

Advancing Genetic Programming for Learning Scheduling Heuristics

by

Meng Xu

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2024

Abstract

Dynamic flexible job shop scheduling (DFJSS) has attracted a lot of attention from both academics and industries because of its widespread industrial impact in the real world. The main challenge in DFJSS involves simultaneous machine assignment and operation sequencing under dynamic environments such as new jobs arriving over time. Among the existing methods, scheduling heuristics stand out as effective approaches widely employed because of their simplicity and real-time responsiveness. Nevertheless, designing scheduling heuristics demands domain knowledge and is a time-consuming process. Genetic Programming (GP), as a hyper-heuristic method, has demonstrated notable success in automatically evolving scheduling heuristics for DFJSS. However, there is room for further enhancement in certain aspects to optimise GP's performance for DFJSS, particularly concerning diversity-based parent selection mechanisms, joint decision-making mechanisms, collaborative heuristic generation and selection by GP and reinforcement learning (RL), and multi-objective problem-solving capabilities.

The overall goal of this thesis is to enhance the effectiveness of GP in automatically evolving high-quality scheduling heuristics for solving the DFJSS problems. Various machine learning and evolutionary computation techniques, such as lexica selection, ensemble, semantic, RL, and multi-objective optimisation, are incorporated into GP to further improve the effectiveness of GP for DFJSS.

First, three novel diversity-based parent selection mechanisms are designed to enhance the GP's ability to explore the search space effectively, adapt to changes, and maintain a balance between exploration and ex-

exploitation for the successful evolution of high-quality scheduling heuristics. The three parent selection mechanisms are cluster selection, diverse partner selection, and lexicase selection. The new mechanisms consider not only the fitness of selected parents but also their diverse strengths, which is crucial for generating high-quality offspring. Experimental results demonstrate that the proposed methods, incorporating these newly designed diversity-based parent selection mechanisms, successfully enhance the quality of scheduling heuristics while promoting increased population diversity.

Second, this thesis proposes an innovative ensemble GP method that employs a population comprising both single individuals and ensembles, and allows the evolution of either a single scheduling heuristic for making individual decisions or a group of scheduling heuristics for making joint decisions for solving the DFJSS problem effectively. For creating an ensemble, an ensemble construction and selection strategy is developed for carefully choosing diverse and complementary individuals from the population. Additionally, new crossover and mutation operators are designed to generate offspring from the selected parents, which can be single individuals or ensembles. By maintaining a population that includes both single individuals and ensembles, the proposed method enables effective exploration of the search space through interbreeding. The experiment results show that the proposed method outperforms the compared algorithms including ensemble GP methods in terms of learned scheduling heuristics.

Third, in addition to the widely used GP for automatic scheduling heuristics learning in DFJSS, RL has gained attention in recent years within this domain. This thesis proposes a niching GP-assisted RL method, taking advantage of both GP and RL to effectively address the DFJSS problems by intelligently heuristic generation by GP and selection by RL simultaneously. Specifically, instead of relying on manual scheduling heuristics, RL actions are replaced with scheduling heuristics evolved by the

niching GP. RL is then employed to optimise and adapt these heuristics based on real-time feedback from the environment. The experiment results demonstrate the effectiveness of the proposed method in comparison to the widely used manual scheduling heuristics and the baseline RL method. This confirms the capability of GP to enhance RL's performance in addressing the DFJSS problem.

Last, this thesis proposes two multi-objective GP methods for evolving a Pareto front of scheduling heuristics for solving the multi-objective DFJSS problem. The first method fills the gap by combining a well-known multi-objective evolutionary algorithm based on decomposition with GP, resulting in a novel multi-objective GP based on the decomposition method. This method successfully enhances the spreadability/diversity of the evolved Pareto front of scheduling heuristics. The second method enhances an existing state-of-the-art multi-objective GP approach (NSGP-II) by incorporating semantic information. The experiment results demonstrate that NSGP-II considering semantic diversity yields better performance compared with the original NSGP-II. Moreover, NSGP-II incorporating semantic similarity achieves even better performance, highlighting the importance of maintaining a reasonable semantic distance between offspring and their parents.

List of Tables

1.1	The characteristics of job information and decision requirements in different types of JSS problems.	2
3.1	The description of terminals and functions.	95
3.2	The parameter settings of GP.	96
3.3	The mean (standard deviation) of the test performance of the 30 independent runs of GP7, GPCS, GPDPS, GPLS on 8 scenarios.	100
3.4	The mean (standard deviation) of the test performance of the 30 independent runs of GP7, GP4, GPLS ⁱ , GPLS on 8 scenarios.	101
3.5	The mean (standard deviation) of the test performance of the 30 independent runs of GPm, STS ^r , STS ^s , GPLS on 8 scenarios.	103
3.6	The mean (standard deviation) of the size of the evolved best routing rules of 30 independent runs of GP7 and GPL for the 8 scenarios.	111
3.7	The mean (standard deviation) of the size of the evolved best sequencing rules of 30 independent runs of GP7 and GPLS for the 8 scenarios.	112
4.1	The configuration of parameters for GP methods.	135

4.2	The mean and standard deviation test performance of the EGP ^e method with different numbers of ensembles in the population through 30 independent runs across 6 scenarios.	138
4.3	The mean and standard deviation performance of the EGP ^e and compared methods on unseen test instances through 30 independent runs across 6 scenarios.	139
4.4	The mean and standard deviation of test performance from 30 independent runs of EGP ^t and EGP ^e across 6 scenarios. .	141
4.5	The mean and standard deviation of test performance from 30 independent runs of EGP ^c and EGP ^e across 6 scenarios. .	142
4.6	The mean (standard deviation) training time (in minutes) of 30 independent runs of GP and EGP ^e methods for 6 scenarios.	153
5.1	The parameter settings of the proposed NichGP method. . .	173
5.2	The parameter configuration of the DRL method.	173
5.3	The mean and standard deviation training performance of 30 independent runs of the proposed NichGPDRL method with different δ values.	176
5.4	The mean and standard deviation test performance of 30 independent runs of the proposed NichGPDRL methods and comparison methods.	178
5.5	The mean and standard deviation behaviour difference of 30 independent runs of the proposed NichGPDRL and GP-DRL methods.	182
5.6	The mean and standard deviation test performance of the DRL, the NichGPDRL, and comparison methods on eight scenarios with a large number of jobs arrival (2000 unit times).	183
5.7	The mean and standard deviation test performance of the DRL, the NichGPDRL, and comparison methods on eight scenarios with a large number of jobs arrival (5000 unit times).	185

5.8	The mean and standard deviation test performance of the DRL, the NichGPDL, and comparison methods on eight scenarios with a large number of workcenters (6 workcenters).	186
5.9	The mean and standard deviation test performance of the DRL, the NichGPDL, and comparison methods on eight scenarios with a large number of workcenters (9 workcenters).	187
6.1	The baseline rules for fitness normalisation.	201
6.2	An example of calculating the PC of an individual.	204
6.3	Six scenarios.	208
6.4	The mean (standard deviation) test HV of 30 independent runs of NSGP, MOGP/D, NSGP ^d and NSGP ^s with different α for 6 scenarios.	209
6.5	The mean (standard deviation) test IGD of 30 independent runs of NSGP, MOGP/D, NSGP ^d and NSGP ^s with different α for 6 scenarios.	211
6.6	The mean (standard deviation) test MS of 30 independent runs of NSGP, MOGP/D, NSGP ^d and NSGP ^s with different α for 6 scenarios.	212

List of Figures

1.1	An example of resource allocation in JSS. The direction of the arrows and the sequence in which they pass through the machines indicate the order of processing on the machines. .	2
1.2	The outline of this thesis, including the main goals and involved techniques of each chapter, and the connection between chapters in this thesis.	22
2.1	The flowchart of traditional GP.	31
2.2	An example of multi-tree-based representation of a scheduling heuristic for DFJSS.	32
2.3	An example of individuals generated by Full and Grow methods.	34
2.4	The process of crossover.	36
2.5	The process of mutation.	37
2.6	An example of generating high-level heuristic from simple low-level heuristics with GP.	38
2.7	An example of DFJSS decision process.	39
2.8	The general framework of RL [153].	41
2.9	An example of end-to-end neural network-based representation of a scheduling heuristic for DFJSS.	42
2.10	The common approaches used for JSS and their classifications.	46
3.1	The flowchart of the proposed GPCS method.	79

3.2	An example of the <i>decision priorities</i> on 3 sequencing decision points and 3 routing decision points.	80
3.3	The flowchart of the GPDPS method for DFJSS.	84
3.4	An example of dividing 8 jobs in a simulation into 4 groups.	87
3.5	An example of the new DPS method.	90
3.6	The flowchart of the proposed GPLS algorithm.	91
3.7	The heat map of test performance of GPLS with different numbers of case and pool size on eight scenarios.	98
3.8	The convergence curves of the test performance of the compared methods on the 8 scenarios.	102
3.9	The violin plots of the standard fitness of the selected parents of GP7, GPm, and GPLS at generation 5, 20, 35, and 45 in scenarios $\langle F_{\max}, 0.85 \rangle$ of a single run.	105
3.10	The convergence curves of unique PCs of GPLS and comparison methods on eight scenarios.	106
3.11	The standard fitness and case-fitness of the top 400 individuals of GPLS at generation 1, 25 and 45 on the $\langle F_{\text{mean}}, 0.85 \rangle$	107
3.12	The standard fitness and case-fitness of the top 400 individuals of GPLS at generation 1, 25, and 45 on the $\langle F_{\max}, 0.85 \rangle$	108
3.13	The Pearson correlation between standard fitness and case-fitness of the top 400 individuals of GPLS at generation 1, 25 and 45 on the $\langle F_{\max}, 0.85 \rangle$ and $\langle F_{\text{mean}}, 0.85 \rangle$	109
3.14	The frequency of the cases used by GPLS at generations 5, 20, 35, and 45 in the $\langle F_{\max}, 0.85 \rangle$ scenario.	110
3.15	An example of the evolved routing rule by GPLS on scenario $\langle F_{\max}, 0.85 \rangle$	113
3.16	An example of the evolved sequencing rule by GPLS on scenario $\langle F_{\max}, 0.85 \rangle$	114
4.1	The flowchart of the proposed EGP ^e method.	122

4.2	The population composition and possible parent combination(s) for crossover and mutation of the proposed EGP ^e method.	124
4.3	The crossover process between an individual and an ensemble.	130
4.4	The crossover process between two ensembles.	131
4.5	The mutation process for an ensemble parent.	132
4.6	The violin plots of the test performance of 30 runs of the proposed EGP ^e with different ratios of single individuals and ensembles on 6 scenarios.	137
4.7	The convergence curves of the mean test performance of 30 runs of standard GP, DivNichGP, and EGP ^e on 6 scenarios. . .	141
4.8	The scatter plots of the train performance of the ensemble versus that of its each element from the number of evaluations 40000 to 50000 of 30 runs by EGP ^e on 6 scenarios. . . .	143
4.9	The scatter plots of the test performance of the ensemble versus the test performance of its each element from the number of evaluations 40000 to 50000 of 30 runs by EGP ^e on 6 scenarios.	144
4.10	The convergence curves of the percentage of the ensemble is output as the best solution every 1000 evaluations on 6 scenarios of 30 runs by the proposed EGP ^e	146
4.11	The number of runs that ensembles are significantly better and worse than, or comparable to individuals out of 30 runs every 1000 evaluations on 3 scenarios of EGP ^e	147
4.12	The fitness distribution of ensembles and individuals of a single run at the beginning (number of evaluations 1000), middle (number of evaluations 25000), and late (number of evaluations 50000) stages of evolution on the scenario $\langle F_{\max}, 0.95 \rangle$ of EGP ^e	148

4.13	The scatter plots of training ensemble contribution versus test ensemble contribution of each individual in the learned ensembles from the number of evaluations 40000 to 50000 by the proposed EGP ^e on 6 scenarios of 30 runs.	151
4.14	The convergence curves of the phenotypic diversity of the individuals in the population by the proposed EGP ^e and EGP ^t on 6 scenarios of 30 runs.	152
4.15	The tree structure of routing rule of an evolved ensemble with its training fitness and training ensemble contribution of a single run on scenario <Fmax, 0.85> of EGP ^e	156
5.1	The overall framework of the proposed NichGPDRL method.	163
5.2	The flowchart of the GP method of Stage 1.	165
5.3	The flowchart of the DRL method of Stage 2.	168
5.4	The box plots of the test performance of 30 independent runs of the proposed NichGPDRL, GPDRL, NichGP#, and DRL methods.	179
5.5	The pie plots of the action contribution of 30 independent runs of the DRL, GPDRL, and the proposed NichGPDRL methods on four scenarios.	180
5.6	The sequencing rule 1 tree structure of four sequencing rules evolved by a NichGP run.	188
5.7	The sequencing rule 2 tree structure of four sequencing rules evolved by a NichGP run.	189
5.8	The sequencing rule 3 tree structure of four sequencing rules evolved by a NichGP run.	189
5.9	The sequencing rule 4 tree structure of four sequencing rules evolved by a NichGP run.	190
6.1	The flowchart of MOGP/D.	199

6.2	The flowchart of the proposed NSGP _{II} with the semantic diversity strategy or the semantic similarity strategy for evolution.	203
6.3	Visualisations of the semantics of individuals of one run of NSGP _{II} , NSGP _{II} ^d , and NSGP _{II} ^s on the scenario 4 and scenario 6 during the start (generation 1), middle (generation 25), and late (generation 50) stages of evolution.	214

List of Publications

- [1] **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “Genetic programming and reinforcement learning on learning heuristics for dynamic scheduling: A preliminary comparison”. *IEEE Computational Intelligence Magazine*, 2024, 15 pp. DOI: 10.1109/MCI.2024.3363970.
- [2] **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “Genetic programming for dynamic flexible job shop scheduling: Evolution with single individuals and ensembles”. *IEEE Transactions on Evolutionary Computation*, 2023, 15 pp. DOI: 10.1109/TEVC.2023.3334626.
- [3] **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “Genetic programming with lexicase selection for large-scale dynamic flexible job shop scheduling”. *IEEE Transactions on Evolutionary Computation*, 2023, 15 pp. DOI: 10.1109/TEVC.2023.3244607.
- [4] **Meng Xu**, Yi Mei, Shiqiang Zhu, Beibei Zhang, Tian Xiang, Fangfang Zhang, and Mengjie Zhang. “Genetic programming for dynamic workflow scheduling in fog computing”. *IEEE Transactions on Services Computing*, vol. 16, no. 4, 2023, pp. 3657–2671.
- [5] **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “Niching Genetic Programming to Learn Actions for Deep Reinforcement Learning in Dynamic Flexible Scheduling”. *IEEE Transactions on Evolutionary Computation*, 2024, 15pp. DOI: 10.1109/TEVC.2024.3395699.

- [6] **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “A semantic genetic programming approach to evolving heuristics for multi-objective dynamic scheduling”. in *Proceedings of the Australasian Joint Conference on Artificial Intelligence*, Springer, 2023, pp. 403–415.
- [7] **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “Multi-objective genetic programming based on decomposition on evolving scheduling heuristics for dynamic scheduling”. in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ACM, 2023, pp. 427–430.
- [8] **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “Genetic programming with diverse partner selection for dynamic flexible job shop scheduling”. in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ACM, 2022, pp. 615–618.
- [9] **Meng Xu**, Fangfang Zhang, Yi Mei, and Mengjie Zhang. “Genetic programming with multi-case fitness for dynamic flexible job shop scheduling”. in *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE, 2022, 8 pp. DOI: 10.1109/CEC55065.2022.9870340.
- [10] **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “Genetic programming with cluster selection for dynamic flexible job shop scheduling”. in *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE, 2022, 8 pp. DOI: 10.1109/CEC55065.2022.9870431.
- [11] **Meng Xu**, Fangfang Zhang, Yi Mei, and Mengjie Zhang. “Genetic programming with archive for dynamic flexible job shop scheduling”. in *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE, 2021, pp. 2117–2124.

Acknowledgments

I would like to express my heartfelt appreciation to my supervisors, A/Prof. Yi Mei, Prof. Mengjie Zhang, and Dr. Fangfang Zhang, for their unwavering guidance and support during my PhD journey. Prof. Mei spends a lot of time and effort helping me to improve my research skills. His dedication to my academic and professional development has left a lasting impact on me, and I am truly fortunate to have had the opportunity to work under his supervision. Prof. Zhang always encourages and guides me during my whole PhD study. His insightful feedback, constructive criticism, and patient guidance have not only refined my scholarly work but have also instilled in me a deeper understanding of my field. I am grateful for Fangfang's kindness and insightful discussions, which have been both useful and enjoyable. I refer to her as Fangfang rather than Dr. Zhang because she has become more like a close friend and sister to me. She not only assists me with my research but also cares deeply about my well-being, offering support during times of stress and uncertainty. I extend my sincere thanks to Prof. Bing Xue, A/Prof. Hui Ma, A/Prof. Aaron Chen, Dr. Qi Chen, Dr. Andrew Lensen, and Dr. Hoai Bach Nguyen for organising meaningful events and group meetings, as well as for their excellent and insightful discussions, which have greatly contributed to my research progress. Additionally, I am thankful to my friend, Dr. Guanqian Gao, for guiding me to this excellent research group.

I am indebted to the China Scholarship Council and the Victoria University of Wellington Scholarship for their financial support throughout

my PhD journey. Furthermore, I would like to acknowledge the Marsden Fund of the New Zealand Government under Contract MFP-VUW1913 and the MBIE SSIF Fund under Contract VUW RTVU1914 for their support in various aspects of my PhD study, including conference attendance.

I am grateful to my friends in the Evolutionary Computation and Machine Learning Research Group for fostering an inspiring research environment and providing companionship. Many thanks to my friends and roommates, Yifan Yang and Zhixing Huang, for creating a supportive atmosphere that enables me to focus on my studies and personal growth. Special thanks to my bestie, Yifan Yang, for her unwavering love, support, and company. She has always been there for me whenever I encounter challenges. Her friendship, sincerity, and shared experiences have enriched my journey in countless ways. I would like to express my gratitude to Qinglan Fan, Peng Wang, Ruwang Jiao, Junhao Huang, Shaolin Wang, Cuie Yang, Tao Shi, Boxiong Tan, Dylon Zeng, Qinyu Wang, Chunyu Wang, Ziyi Sun, Hengzhe Zhang, Jordan MacLachlan, and many others for their warm friendship and companionship. Additionally, I am thankful to Gonglin Yuan, Wenbin Pei, and Ying Bi for their guidance in my career development and for sharing their valuable experiences.

My deepest gratitude goes to my parents for their unconditional love, sacrifices, and unwavering belief in my abilities. Their support and encouragement have been my pillar of strength through life's challenges, and I am forever grateful for everything they have done for me.

Last but not least, I want to express my deepest appreciation to my boyfriend, Sifan Zhang, for his love, encouragement, and understanding. His support and belief in me have been my greatest source of strength and motivation. Thank you for being my rock and my biggest cheerleader.

Contents

1	Introduction	1
1.1	Job Shop Scheduling	1
1.2	Motivations	5
1.2.1	Fundamental Challenges of GP for DFJSS	5
1.2.2	Limitations of Existing Parent Selection	6
1.2.3	Limitations of Existing Ensemble GP	8
1.2.4	Collaborative Intelligent Heuristic Generation and Selection through the Integration of GP and RL	10
1.2.5	Limitations of Existing Multi-objective GP	12
1.3	Research Goals	13
1.4	Major Contributions	16
1.5	Terminology	20
1.6	Organisation of Thesis	21
2	Background	25
2.1	Dynamic Flexible Job Shop Scheduling	25
2.2	Basic Concepts and Approaches	28
2.2.1	Machine Learning	28
2.2.2	Evolutionary Computation	29
2.2.3	Genetic Programming	31
2.2.4	Heuristic and Hyper-heuristic	37
2.2.5	Scheduling Heuristics	39
2.2.6	Reinforcement Learning	40

2.3	Job Shop Scheduling Approaches	45
2.3.1	Exact Approaches	45
2.3.2	(Meta-)Heuristic Approaches	47
2.3.3	Hyper-heuristic Approaches	49
2.4	GP for Evolving Scheduling Heuristics	51
2.4.1	Static Scheduling Problems	52
2.4.2	Dynamic Scheduling Problems	54
2.5	Related Work	59
2.5.1	Parent Selection in GP	59
2.5.2	Ensemble GP	63
2.5.3	RL for JSS	66
2.5.4	Multi-objective GP	69
2.6	Chapter Summary	72
3	Diversity-based Parent Selection Mechanisms	75
3.1	Introduction	75
3.1.1	Chapter Goals	77
3.1.2	Chapter Organisation	78
3.2	Proposed Algorithms	78
3.2.1	GP with Cluster Selection	78
3.2.2	GP with Diverse Partner Selection	83
3.2.3	GP with Lexicase Selection	90
3.3	Experimental Design	94
3.3.1	Dataset	94
3.3.2	Parameter Setting	95
3.3.3	Comparison Design	96
3.4	Results and Discussions	98
3.4.1	Sensitivity Analysis	98
3.4.2	Test Performance	99
3.4.3	Distribution of Parent Fitness	104
3.4.4	Phenotypic Diversity	105

3.5	Further Analyses	106
3.5.1	Correlation between Case-Fitnesses	106
3.5.2	Frequency of Case Usage	110
3.5.3	Rule Size	111
3.5.4	Insight on the Evolved Scheduling Heuristics	112
3.5.5	Further Discussions	115
3.6	Chapter Summary	116
4	Joint Decision-making by Ensemble	119
4.1	Introduction	119
4.1.1	Chapter Goals	120
4.1.2	Chapter Organisation	122
4.2	Proposed Algorithm	122
4.2.1	Overall Framework	122
4.2.2	Population Initialisation	124
4.2.3	Individual Evaluation	125
4.2.4	Ensemble Construction and Selection	126
4.2.5	Ensemble Evaluation	128
4.2.6	Genetic Operators	129
4.2.7	Computational Complexity	132
4.3	Experimental Design	134
4.3.1	Parameter Setting	134
4.3.2	Comparison Design	134
4.4	Results and Discussions	136
4.4.1	Sensitivity Analysis	136
4.4.2	Test Performance	138
4.4.3	Effectiveness of Ensemble Construction	140
4.4.4	Effectiveness of Genetic Operators	142
4.5	Further Analyses	142
4.5.1	Performance of Ensemble and Elements	142
4.5.2	Ensemble Win Percentage	145

4.5.3	Elements Contribution to Ensemble	149
4.5.4	Diversity	150
4.5.5	Training Time	153
4.5.6	Structure Analyses of Elements in Ensemble	154
4.6	Chapter Summary	157
5	Collaborative Heuristic Generation and Selection by Genetic Programming and Reinforcement Learning	159
5.1	Introduction	159
5.1.1	Chapter Goals	161
5.1.2	Chapter Organisation	162
5.2	Proposed Algorithm	162
5.2.1	Overall Framework	162
5.2.2	State Features	163
5.2.3	Stage 1: Niching GP training	165
5.2.4	Stage 2: DRL training	167
5.3	Experimental Design	170
5.3.1	Dataset	170
5.3.2	Parameter Setting	172
5.3.3	Comparison Design	174
5.4	Results and Discussions	176
5.4.1	Influence of the Radius Parameter in NichGP	176
5.4.2	Test Performance	177
5.5	Further Analyses	181
5.5.1	Action Contribution	181
5.5.2	Behaviour Difference between Heuristics	181
5.5.3	Generalisation to More Complex Scenarios	182
5.5.4	Discussions on Enhancing GP capability	184
5.5.5	Structure Analysis of Sequencing Rules	186
5.6	Chapter Summary	192

6	Multi-objective Genetic Programming	195
6.1	Introduction	195
6.1.1	Chapter Goals	197
6.1.2	Chapter Organisation	198
6.2	Proposed Algorithms	198
6.2.1	MOGP/D	198
6.2.2	Semantic NSGPII	201
6.3	Experimental Design	206
6.3.1	Performance Measures	206
6.3.2	Comparison Design	207
6.4	Results and Discussions	208
6.4.1	Test Performance	208
6.5	Further Analyses on Population Distribution	214
6.6	Chapter Summary	215
7	Conclusions	217
7.1	Achieved Objectives	217
7.2	Main Conclusions	220
7.2.1	Diversity-based Parent Selection Mechanisms	220
7.2.2	Joint Decision Making with Ensemble	221
7.2.3	Intelligent Heuristic Generation and Selection by GP and RL	223
7.2.4	Multiple Objectives-solving Ability in GP	224
7.2.5	Overall Contributions to GP and AI	225
7.3	Future Work	226
7.3.1	Training Time Reduction	226
7.3.2	Interpretability of Scheduling Heuristics	228
7.3.3	Effective Utilisation of Multiple Scheduling Heuristics	229
7.3.4	Advanced Multi/many-objective Optimisation	231
7.3.5	Integration of GP and RL	231
	Bibliography	235

Chapter 1

Introduction

1.1 Job Shop Scheduling

Job shop scheduling (JSS) [29] is one of the most prominent problems in industrial production, and it has received extensive attention from scholars and industries. As an important combinatorial optimisation problem, JSS has the purpose of optimising resource allocation [67], as illustrated in Figure 1.1. JSS has many applications in practical fields, such as automotive assembly [280], textile manufacturing [223], chemical material processing [241], and semiconductor manufacturing [71]. JSS can be classified into various categories based on its characteristics. Table 1.1 outlines the distinctive features of job information and decision requirements across different types of JSS. Subsequently, this thesis offers a brief description of each category.

Classical Static Job Shop Scheduling: Classical static JSS [29] involves constraints determining the operation sequence of multiple jobs, including dependencies among operations within the same job. On the shop floor, there exists a set of machines, and numerous jobs await processing. Each job comprises a number of operations, and an operation can only be processed after the completion of its preceding operation. Additionally, each operation can only be assigned to a fixed machine without flexibility.

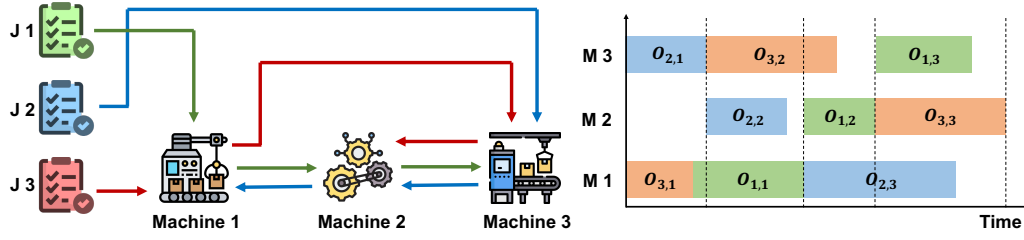


Figure 1.1: An example of resource allocation in JSS. The direction of the arrows and the sequence in which they pass through the machines indicate the order of processing on the machines.

Table 1.1: The characteristics of job information and decision requirements in different types of JSS problems.

	Problem	Static JSS	Dynamic JSS	Flexible JSS	DFJSS
Job information	Known	✓		✓	
	Unknown		✓		✓
Decision	Sequencing	✓	✓	✓	✓
	Routing			✓	✓

Consequently, the objective is to determine the sequence of operations on machines to minimise the makespan, reduce production costs, or achieve other specified objectives.

Flexible Job Shop Scheduling: Flexible JSS [233, 179] extends the concept of a traditional job shop by introducing flexibility in machine assignments [233]. In flexible JSS, operations are not restricted to a single dedicated machine; instead, they can be processed on a set of optional machines [81]. Consequently, decision-making in flexible JSS involves not only selecting the next operation when a machine is available (i.e., sequencing decision point) but also determining the optimal machine to handle a ready operation (i.e., routing decision point) [38]. It is worth noting that flexible JSS is recognised as an NP-hard problem [38].

Dynamic Job Shop Scheduling: For *static* JSS, job information is known in advance, allowing scheduling decisions to be made with com-

plete knowledge of all job details. However, real-world scheduling processes are often influenced by dynamic events such as the arrival of new jobs, order cancellations or modifications, uncertainties in processing times, and machine breakdowns. The dynamically changed system state makes the problem more complex and makes it more relevant for practical applications. The JSS problem with dynamic change characteristics is known as the dynamic JSS problem [115, 197, 242, 251, 263].

Dynamic Flexible Job Shop Scheduling: Dynamic flexible JSS (DFJSS) combines the challenges of both dynamic JSS and flexible JSS. It holds a significant appeal within manufacturing systems and has been intensively studied in previous decades. DFJSS needs to address both machine flexibility and dynamic events simultaneously [263]. Among dynamic events, the most frequent and common event in the real world is the dynamic arrival of new jobs [268]. Therefore, in this thesis, we focus on DFJSS with new dynamic job arrivals.

Methods for solving JSS problems can be broadly categorised into different groups based on their characteristics. In the early stage, *exact methods* (e.g., mathematical programming [81]) and *heuristic methods* (e.g., Tabu search [190], simulated annealing [67], and particle swarm optimisation [181]) have been proposed for solving JSS problems. While exact methods are well-suited for small-scale JSS problems, their efficacy diminishes for large-scale JSS and dynamic JSS problems with real-time requirements. Similarly, heuristic methods face limitations in adapting to real-time requirements, making them unsuitable for solving dynamic JSS problems.

In response to this, **scheduling heuristics**, employed as priority functions, have gained widespread use in solving dynamic JSS problems [265]. Their reusable nature, capacity to leverage the latest information, and real-time responsiveness make them particularly effective and practical. In the context of JSS, a scheduling heuristic typically involves a single sequencing rule, used to select an operation at the sequencing decision point. While in flexible JSS, where both operation sequencing and machine as-

signment (routing) need to be considered, a scheduling heuristic commonly comprises two rules: a sequencing rule and a routing rule [258]. The sequencing rule remains responsible for selecting an operation at the sequencing decision point, while the routing rule aids in machine selection when the routing decision point is encountered. Numerous empirical studies [97, 135, 145, 159] have demonstrated the effectiveness of scheduling heuristics in addressing JSS problems.

Although scheduling heuristics have the advantage of generating schedules in real-time, their design demands domain knowledge and is time-consuming [259]. Additionally, no single scheduling heuristic proves universally effective for diverse customer requirements. In the real world, there are many scenarios that need to be considered for DFJSS. Therefore, an automatic design of scheduling heuristics for different scenarios is needed. *Hyper-heuristic* methods [30] have been employed to generate high-level heuristics by selecting or combining a set of low-level heuristics to address complex problems.

Genetic Programming (GP) [119], as a hyper-heuristic method, has been widely used to automatically evolve scheduling heuristics that produce high-quality schedules for DFJSS [258]. Among the various evolutionary computational approaches to solving the DFJSS problems, GP has several key advantages that set it apart from others and are noteworthy. First of all, GP has a flexible representation, e.g. tree-based GP, which makes it easy to explore the structure and features with little domain knowledge [258]. Secondly, the scheduling heuristics evolved by GP offer good interpretability, which provides more confidence to users to apply these scheduling heuristics in real-world applications [265]. Finally, the generalisation ability of these scheduling heuristics evolved through GP is much stronger compared to other evolutionary computation approaches [263]. Existing studies [3, 114, 144, 146, 156, 252, 268] have shown the advantages of GP. In this case, this thesis focuses on improving the effectiveness of GP in solving the DFJSS problems.

1.2 Motivations

This section begins with an overview of the fundamental challenges involved with DFJSS. It then presents insights into the rationale behind the adoption of GP as the main method for solving the DFJSS problems. Afterwards, the motivations behind the four specific contributions in this thesis are introduced.

1.2.1 Fundamental Challenges of GP for DFJSS

DFJSS is a challenging problem owing to the following reasons:

1. **Dynamic nature and real-time adaptation [84]:** DFJSS involves scheduling jobs with multiple operations in a manufacturing environment where the information of jobs is unknown in advance. This dynamic nature introduces uncertainties and the need for real-time adaptation in the production environment makes it challenging to create good and adaptable schedules.
2. **Requirements of diverse solutions in large search space [124]:** The number of possible combinations and sequences for scheduling operations of multiple jobs across multiple machines and decision points grows exponentially with the problem size. This leads to a vast search space, making it computationally challenging to find optimal solutions. To handle this complexity, there is a crucial need for diverse solutions to prevent getting trapped in local optima and enable the exploration of a broader solution landscape.
3. **Good, stable, and trustworthy decisions requirements [209]:** Achieving high-quality schedules in dynamic environments requires decisions that are not only good concerning traditional scheduling objectives but also stable and trustworthy. The schedules must be robust enough to withstand disruptions caused by dynamic events.

Stability ensures that schedules remain effective even as the system changes. Trustworthiness gives customers confidence in believing the decisions made in dynamic scenarios where unexpected events may occur. This leads to a requirement of joint decision making by using a number of scheduling heuristics.

4. Large number of decision points requiring intelligent strategies:

The dynamic and flexible nature of DFJSS introduces a large number of decision points, including machine assignments and operation sequencing. The complexity upgrades as the jobs continue to arrive. Intelligent decision-making (e.g., intelligent heuristic generation and selection) becomes essential to navigate this intricate decision space. Algorithms must adapt in real-time, making informed decisions to optimise resource usage and scheduling performance. The challenge is to develop algorithms that can handle the increased decision points, ensuring adaptability and efficiency in a dynamically changing environment.

5. Multi-objective optimisation [225]: DFJSS often involves optimising multiple potential conflicting objectives simultaneously, such as minimising makespan, reducing production costs, and meeting delivery deadlines. Balancing these objectives to achieve a well-rounded solution is inherently challenging.

1.2.2 Limitations of Existing Parent Selection

Parent selection determines which individuals from the current population will contribute genetic material to produce offspring for the next generation, playing an important role in influencing the evolutionary process and the quality of offspring generated [228]. Its importance lies in several key aspects. Firstly, an effective parent selection mechanism helps preserve valuable building blocks [269]. This aids in maintaining important genetic material over generations, enhancing the algorithm's ability

to evolve high-quality solutions. Secondly, an effective parent selection mechanism promotes diversity within the population [191]. A diverse population is essential for preventing premature convergence [49]. By selecting parents strategically, the algorithm can maintain a diverse set of individuals, fostering a richer exploration of the search space.

The most commonly employed parent selection for GP is tournament selection [62]. This approach involves randomly selecting a set of candidate individuals and then choosing the candidate individual with the best fitness as the parent. Tournament selection is applicable for crossover, mutation, and reproduction. While mutation and reproduction require only one parent to generate offspring, crossover needs two parents working together. However, the independent selection of the two parents for crossover through tournament selection may lead to similarities in their behaviour, which is not good for generating high-quality offspring with diverse building blocks. This observation motivates the exploration of new parent selection mechanisms capable of not only identifying high-quality individuals based on fitness but also ensuring diverse behaviours in the selected parents.

Additionally, tournament selection faces the challenge that the chosen parents may not have effective cooperative abilities for generating high-quality offspring. In response to this limitation, comparative partner selection [50] and diverse partner selection [7] have been designed to select pairs of complementary parents. However, these methods cannot be directly employed to address the DFJSS problems. This motivates us to develop a novel parent selection mechanism capable of selecting individuals with strong cooperative abilities to produce high-quality offspring specifically tailored for DFJSS.

Furthermore, both of these methods are specifically tailored for crossover and do not consider mutation. Although mutation occurs less frequently than crossover, it still plays a role in influencing the evolutionary process and solution quality. Lexicase selection [147] was proposed

to select parents by following different orders of training cases, making it suitable for both crossover and mutation [2, 121, 122]. Studies have demonstrated that lexicase selection can pick more diverse parents and yield better performance than tournament selection in certain problems [123, 152, 168, 203]. Thus, it is reasonable to extend the GP with lexicase selection to evolve scheduling heuristics as well. However, it is non-trivial to extend the GP with lexicase selection from its successful domains such as program synthesis and symbolic regression to evolving scheduling heuristics, since the evaluation of an instance in DFJSS is very time-consuming. In DFJSS, an instance is usually a large-scale DFJSS simulation with thousands of jobs. Evaluating such an instance requires applying the GP model many times (each at a decision situation, when a machine becomes idle or a job operation becomes ready, to be processed) to generate a schedule given the dynamic job arrivals in the simulation. As a result, we cannot afford a large number of DFJSS instances (i.e., simulations). On the other hand, if we use too few instances, then the effectiveness of lexicase selection might be negatively affected. This motivates us to develop a GP with a new lexicase selection mechanism that can apply lexicase selection effectively in GP for DFJSS without increasing complexity.

Therefore, there is a need for effective parent selection mechanisms capable of identifying diverse individuals with the potential to enhance the quality of scheduling heuristics or complement each other effectively in generating high-quality offspring in GP for DFJSS.

1.2.3 Limitations of Existing Ensemble GP

Relying on a single scheduling heuristic may not yield optimal performance across all decision points and scenarios. Also, the confidence in decisions made by using only one scheduling heuristic may be unsatisfactory [72]. **Joint decision-making** involving a group of scheduling heuristics has the potential to enhance overall performance and provide more

reliable decisions. The ensemble is a promising technique that considers cooperation between elements to make final decisions [177, 74]. To be specific, ensemble GP can leverage a group of different programs to obtain the final solution and improve the credibility of the decision [187, 75].

Several studies [78, 88, 205, 211] have explored the application of ensemble GP to address JSS problems. The existing literature on ensemble GP for scheduling problems can be categorised into three groups. The *first* category involves the use of multiple subpopulations to simultaneously evolve scheduling heuristics, followed by grouping the best ones from each subpopulation to form an ensemble [176]. While this approach produces high-quality scheduling heuristics, it often overlooks cooperation between them during evolution, considering their evolution as independent processes. The *second* category employs a single population to evolve scheduling heuristics, selecting a subset from the final generation to constitute the ensemble [222]. This method adheres to conventional evolutionary processes, considering diversity through mechanisms like niching, or evolving based on different subsets of training instances to generate a diverse set of effective scheduling heuristics. However, despite addressing diversity concerns, this category ignores cooperation between scheduling heuristics, with the evolution of each heuristic proceeding independently. The *third* category explores the evolution of both single individuals and ensembles [187]. Nonetheless, the evolution of single individuals and ensembles remains independent. For instance, this category typically evolves a set of scheduling heuristics using GP and then employs heuristic methods (e.g., genetic algorithms) to evolve an ensemble. In such cases, direct contributions between single individuals and ensembles during the evolution process, which could enable mutual benefit, are limited.

In summary, current studies on ensemble GP for solving scheduling problems are at an early stage and require further exploration. A novel ensemble GP method is essential for DFJSS, capable of identifying and leveraging the strengths of both single individuals and ensembles, foster-

ing mutual benefit between them.

1.2.4 Collaborative Intelligent Heuristic Generation and Selection through the Integration of GP and RL

GP has long been a successful approach to learning effective scheduling heuristics in DFJSS [262]. GP employs evolutionary principles to learn scheduling heuristics through an iterative, population-based, and stochastic process [35]. The key advantage of using GP for DFJSS lies in its ability to automatically explore a wide range of potential scheduling heuristics by maintaining a population and adapt to the specific characteristics of the problem at hand with little domain knowledge [206]. Current approaches in GP for DFJSS commonly involve intelligently heuristic generation, evolving either a single scheduling heuristic [284] or a group of heuristics [212] to make decisions across all decision points during the entire scheduling process. However, this generalised use of scheduling heuristics might not fully exploit the unique strengths of each heuristic. Manually identifying and distinguishing the advantages of different scheduling heuristics is challenging and time-consuming. Hence, the integration of an intelligent heuristic selection mechanism becomes crucial. This mechanism is expected to automate the selection process of the most appropriate scheduling heuristic evolved by GP for a given decision point through interactions with the scheduling environment. This way allows for more efficient utilisation of a number of scheduling heuristics evolved by GP in a single run, reducing dependence on a solitary “best” heuristic and minimising the underutilisation of other potentially effective alternatives. Prior to implementing the heuristic selection method, a set of candidate scheduling heuristics is required. These scheduling heuristics should demonstrate high quality and exhibit diverse behaviour. While RL can be a valuable method for generating scheduling heuristics, it typically updates and refines only one heuristic at a time. This way necessitates

multiple runs of the RL algorithm to generate a diverse set of heuristics, which can be inefficient. Traditional GP methods often prioritise obtaining the best scheduling heuristic without emphasising diversity within the population. Niching [246] has proven to be an effective strategy employed in GP to enhance its efficacy [143], achieved by augmenting population diversity and fostering the presence of multiple diverse scheduling heuristics within the population.

Beyond GP, Reinforcement learning (RL) [36], as a subfield of artificial intelligence, has gained increasing attention for learning scheduling agents in DFJSS. RL focuses on creating intelligent decision-making systems through learning from interactions with an environment, taking actions, and receiving feedback [137]. Its intelligent decision-making mechanism is characterised by its adaptive capability to refine policies over time based on experiences and feedback [136]. RL has become popular in training scheduling agents for solving complex scheduling problems. While RL excels in end-to-end (direct) learning processes, some challenges arise in DFJSS scenarios where the number of candidate machines/operations might change dynamically during the scheduling process [136]. This variability makes it challenging to fix action spaces and store experiences in RL. To address this, some RL studies employ manual scheduling heuristics as actions instead of directly using machines/operations [36, 139]. Then RL conducts heuristic selection among these actions. However, this indirect approach presents limitations. Firstly, the manually designed scheduling heuristics often exhibit an average level of quality, thereby constraining the overall quality of high-level scheduling heuristics learned by RL. Secondly, the manual design process diverse scheduling heuristics is time-consuming and requires a lot of domain knowledge from experts.

Considering the strengths and weaknesses of both GP and RL, GP excels at *generating* multiple comprehensive scheduling heuristics simultaneously through population evolution, while RL is good at *selecting* among multiple scheduling heuristics at specific decision points. Therefore, ex-

ploring collaborations between them could be beneficial. The exploration of **intelligent scheduling heuristic generation and selection using a combination of GP and RL** might be an interesting dimension to the existing studies on DFJSS. This integration aims to enhance the capabilities of both GP and RL in effectively solving the DFJSS problems.

1.2.5 Limitations of Existing Multi-objective GP

The study of the **multi-objective DFJSS (MO-DFJSS)** problem is important as it has a significant impact on industrial applications [160]. In real-world applications, scheduling problems usually involve multiple conflicting objectives. For instance, minimising makespan might conflict with meeting delivery deadlines. Optimising multiple objectives simultaneously can balance these conflicting objectives, offering customers a range of trade-off options. In this case, it is necessary to study the MO-DFJSS problem.

Currently, several studies investigate the fusion of Pareto dominance-based algorithms, such as the non-dominated sorting genetic algorithm II (NSGA-II) [52] and the strength Pareto evolutionary algorithm 2 (SPEA2) [289], with GP to address the MO-DFJSS problems [267, 282]. However, Pareto-dominance algorithms might face difficulties in maintaining the spreadability/diversity of the Pareto front. A diverse Pareto front ensures that the solutions cover a wide range of trade-offs between conflicting objectives. This diversity provides users with a variety of options to choose from, allowing them to select the heuristic that best suits their requirements. This limitation of Pareto-dominance algorithms emphasises the necessity for a method capable of ensuring significant spreadability in the evolved Pareto front. The multi-objective evolutionary algorithm based on decomposition (MOEA/D) is a prominent multi-objective method [276]. Its strength lies in the potential to generate a spread/diverse Pareto front by decomposing a multi-objective problem into several sub-problems us-

ing a scalar function and optimising them simultaneously [85]. However, to the best of our knowledge, no study has explored the use of MOEA/D for solving the MO-DFJSS problems. Therefore, it is both interesting and meaningful to investigate whether integrating MOEA/D with GP can facilitate the evolution of a Pareto-front of high-quality and good spreadability scheduling heuristics for solving the MO-DFJSS problems.

Additionally, existing studies indicate that among current multi-objective GP methods for MO-DFJSS, NSGP-II demonstrates superior performance [267]. However, NSGP-II operates solely on the genotype of individuals and does not take into account semantic information, which reflects the behaviour of the genotype. Consequently, NSGP-II might suffer from a loss of population diversity. Maintaining diversity ensures that the algorithm explores a wide range of solutions across the entire Pareto front., facilitating the discovery of novel and potentially superior solutions in less-explored regions of the solution space. Therefore, there is a need to incorporate semantic information into NSGP-II to address this limitation. Semantic GP, as proposed in previous research [217], offers a solution to enhance population diversity by integrating semantic information into the evolutionary process. Its effectiveness has been demonstrated across various domains, including symbolic regression [214], classification [15, 192], and feature selection [169]. However, to the best of our knowledge, semantic information has not been integrated into NSGP-II for solving MO-DFJSS problems. Given the promising results obtained with semantic GP in other domains, it is particularly intriguing to explore whether and how incorporating semantic information can improve the performance of NSGP-II for MO-DFJSS.

1.3 Research Goals

The overall goal of this thesis is to improve the capability of GP for DFJSS by addressing key aspects, including effective parent selection,

joint decision-making, intelligent heuristic usage (selection), and multi-objective problem-solving. Specifically, this thesis focuses on enhancing GP by developing novel diversity-based parent selection mechanisms, refining decision-making processes by adopting an ensemble technique for joint decisions, selecting appropriate scheduling heuristics evolved by GP intelligently by incorporating RL techniques, and handling multiple objectives through the integration of multi-objective optimisation mechanisms. The details of each objective are shown as follows.

Objective 1: Develop GP with **new diversity-based parent selection mechanisms** to select promising and diverse individuals as parents for automatically evolving effective scheduling heuristics for DFJSS.

The proposed methods are hypothesised to enhance scheduling heuristics for DFJSS by developing effective parent selection mechanisms. This thesis proposes three parent selection mechanisms. The first one is the cluster selection, it is expected to select individuals with different behaviours as parents. The second one is the diverse partner selection, it is expected to select individuals with complementary strengths as parents. The final one is the new lexicase selection, it is expected to select expert individuals who are effective for different cases as parents. By selecting such individuals as parents, the goal is to generate high-quality offspring that will, in turn, enhance the efficacy of the evolved scheduling heuristics for addressing the DFJSS problems.

Objective 2: Propose a **novel ensemble** GP method to automatically evolve effective single scheduling heuristics or a group of scheduling heuristics to make effective joint decisions for solving the DFJSS problems.

The proposed method is hypothesised to maintain a population that includes both individuals and ensembles, allowing breeding between them for more effective exploration of the search space. Additionally, a strategy for ensemble construction and selection is developed to form ensembles by emphasising diversity and complementarity among individuals. Furthermore, new crossover and mutation operators are designed to generate

high-quality offspring from both individuals and ensembles.

Objective 3: Propose a two-stage framework that utilises GP with a **niching strategy for learning heuristic actions for RL** to enable intelligent GP heuristic selection in solving the DFJSS problems.

Specifically, an initial comparison is conducted between a typical GP method and a typical RL method for DFJSS, aiming to find their respective strengths and weaknesses. Subsequently, a two-stage framework is proposed. In the first stage, a niching GP is proposed to evolve a set of high-quality and diverse scheduling heuristics, to be employed as actions for RL. In the second stage, an RL method is employed. It is hypothesised to use the high-quality and diverse scheduling heuristics evolved from the niching GP to train intelligent scheduling agents, enabling intelligent heuristic usage at various decision points.

Objective 4: Propose **novel multi-objective** GP methods for MO-DFJSS to automatically evolve a Pareto front of scheduling heuristics to handle multiple objectives simultaneously.

The proposed method is hypothesised to evolve a Pareto front of effective scheduling heuristics capable of handling multiple objectives simultaneously. This thesis proposes two multi-objective GP methods. The first one is the multi-objective GP based on decomposition (MOGP/D) method. It is hypothesised to fill the gap of no existing work incorporating the well-known MOEA/D and GP for solving the MO-DFJSS problems. The second method is the semantic NSGP-II, which incorporates semantic information to reflect the behaviour of individuals' genotypes. Semantic NSGP-II employs two strategies: semantic diversity and semantic similarity. The semantic diversity strategy aims to generate offspring with semantically different behaviours. The semantic similarity strategy is hypothesised to generate offspring that not only exhibit semantic diversity but also maintain similarity with their parents.

1.4 Major Contributions

This thesis makes the following major contributions, each of which is discussed in the respective contribution Chapters (Chapters 3 to 6).

Firstly, this thesis has demonstrated how to evolve effective scheduling heuristics by proposing effective diversity-based parent selection mechanisms to select parents to solve the DFJSS problems.

To be specific, this thesis proposes three parent selection mechanisms, which are cluster selection, diverse partner selection, and novel lexicase selection. Cluster selection contributes to the selection of two parents with different behaviours for crossover and can increase the number of unique behaviours in the population. Diversity partner selection helps in selecting a pair of high-quality and complementary parents for crossover. The results show that the GP with the proposed diverse partner selection outperforms the baseline GP across all maximum objectives and achieves comparable performance on all mean objectives. The novel lexicase selection contributes to the selection of individuals with expertise in various cases as parents. To avoid increasing the computational cost by using lexicase selection, a multi-case fitness evaluation method is proposed. This method addresses the time-consuming evaluation of each instance through a new definition of case-fitness, generating multiple cases from a single simulation. Importantly, the newly developed multi-case fitness definition is not limited to DFJSS but offers general guidelines for efficiently utilising lexicase selection in solving other complex problems with time-consuming fitness evaluations, such as dynamic vehicle routing and cloud resource allocation. Furthermore, the superiority of the proposed GP with the lexicase selection algorithm over other GP algorithms in various DFJSS scenarios highlights the effectiveness of incorporating lexicase selection into GP. This integration enhances population diversity and strikes a better balance between exploration and exploitation, showcasing its potential applicability in addressing a wide range of complex problems.

Part of the contributions have been published in:

- **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “Genetic programming with lexicase selection for large-scale dynamic flexible job shop scheduling”. *IEEE Transactions on Evolutionary Computation*, 2023, 15 pp. DOI: 10.1109/TEVC.2023.3244607.
- **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “Genetic programming with diverse partner selection for dynamic flexible job shop scheduling”. in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ACM, 2022, pp. 615–618.
- **Meng Xu**, Fangfang Zhang, Yi Mei, and Mengjie Zhang. “Genetic programming with multi-case fitness for dynamic flexible job shop scheduling”. in *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE, 2022, 8 pp. DOI: 10.1109/CEC55065.2022.9870340.
- **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “Genetic programming with cluster selection for dynamic flexible job shop scheduling”. in *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE, 2022, 8 pp. DOI: 10.1109/CEC55065.2022.9870431.

Secondly, this thesis has demonstrated how to use GP to evolve high-quality single scheduling heuristics or a group of diverse scheduling heuristics to enable effective joint decision-making in DFJSS.

This is achieved by proposing a novel ensemble GP method. Different from conventional approaches, this method facilitates the evolution between single individuals and ensembles, enhancing flexibility in the evolutionary process and improving exploration of the search space. Experimental results demonstrate the superior performance of the proposed method compared to standard GP and existing ensemble GP methods for solving the DFJSS problems. Within the proposed ensemble GP method, a novel ensemble construction and selection strategy are designed to help the proposed method in selecting diverse and high-quality individuals for

constructing an ensemble. Additionally, new crossover and mutation operators are developed to facilitate evolution between single individuals and ensembles. These operators generate offspring that can be either single individuals or ensembles, fostering more flexible breeding between the two. Experimental results validate the effectiveness of the developed ensemble construction and selection strategy and show the efficacy of the newly designed genetic operators. Further analyses reveal that scheduling heuristics within a promising ensemble exhibit varied behaviour while still sharing certain similarities. This enables them to make consistent decisions in most cases and complement each other at specific decision points.

Part of the contributions have been published in:

- **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “Genetic programming for dynamic flexible job shop scheduling: Evolution with single individuals and ensembles”. *IEEE Transactions on Evolutionary Computation*, 2023, 15 pp. DOI: 10.1109/TEVC.2023.3334626.

Thirdly, this thesis has demonstrated how to combine GP with RL to take advantage of both methods for making intelligent generation and selection of multiple scheduling heuristics for solving the DFJSS problems effectively.

This is achieved through a systematic comparison between a typical GP and a typical RL method for DFJSS, followed by the development of a new two-stage framework based on niching GP and RL. In the initial stage, a niching GP is proposed to automatically evolve high-quality scheduling heuristics, which subsequently serve as actions for RL. The second stage employs RL to intelligently utilise these evolved scheduling heuristics. This method addresses the challenge of adapting to changes in operations at various decision points, enabling the learning of intelligent agents capable of making good selections among these actions to generate effective schedules. As an initial exploration into the integration of GP and RL, this thesis investigates replacing the manual sequencing rules in

baseline RL with GP-evolved ones as actions, while maintaining the end-to-end RL process for learning the routing agent identical to the baseline DRL. Comparative results against baseline RL and commonly used manually designed scheduling heuristics confirm the effectiveness of the proposed method. Additionally, comparisons against traditional GP-assisted RL validate the efficacy of the proposed niching GP method. Furthermore, contrasting results against the proposed method without the RL training process, where the sequencing rule is fixed as one of the candidate actions, verify the effectiveness of the RL learning process.

Part of the contributions have been published in:

- **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “Genetic programming and reinforcement learning on learning heuristics for dynamic scheduling: A preliminary comparison”. *IEEE Computational Intelligence Magazine*, 2023, 15 pp. DOI: 10.1109/MCI.2024.3363970.
- **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “Niching Genetic Programming to Learn Actions for Deep Reinforcement Learning in Dynamic Flexible Scheduling”. *IEEE Transactions on Evolutionary Computation*, 2024, 15pp. DOI: 10.1109/TEVC.2024.3395699.

Finally, this thesis has demonstrated how to use GP for evolving a Pareto front of effective scheduling heuristics for solving multiple objectives in DFJSS simultaneously.

This is achieved by proposing two effective multi-objective GP methods. The first method develops an effective multi-objective GP based on decomposition (MOGP/D) method for MO-DFJSS by adapting ideas from the classical MOEA/D along with the characteristics of MO-DFJSS. To mitigate the impact of seed rotation in DFJSS [266], a mapping strategy is proposed to match individuals to sub-problems after evaluation. The effectiveness of the proposed MOGP/D method in terms of the spreadability/diversity of the evolved Pareto front and consistency of the scheduling

heuristics behaviours is validated by comparing it with the state-of-the-art Pareto-dominance multi-objective method (NSGP-II). However, in terms of HV and IGD, NSGP-II performs better than MOGP/D. The second method is the semantic NSGP-II method, enhancing the NSGP-II by integrating semantic information. This method defines the semantic and semantic distance of scheduling heuristics for DFJSS. Then, by incorporating semantic diversity and semantic similarity within NSGP-II, this method contributes to evolving better scheduling heuristics than using the original NSGP-II. The results highlight the benefits of considering semantically diverse individuals for achieving high-quality scheduling heuristics. Moreover, NSGP-II, considering semantic similarity, achieves the best overall performance, offering valuable insight into the importance of maintaining a reasonable semantic distance between offspring and their parents to further enhance the quality of scheduling heuristics. The results also emphasise the trade-off between semantic diversity and semantic similarity.

Part of the contributions have been published in:

- **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “A semantic genetic programming approach to evolving heuristics for multi-objective dynamic scheduling”. in *Proceedings of the Australasian Joint Conference on Artificial Intelligence*, Springer, 2023, pp. 403–415.
- **Meng Xu**, Yi Mei, Fangfang Zhang, and Mengjie Zhang. “Multi-objective genetic programming based on decomposition on evolving scheduling heuristics for dynamic scheduling”. in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ACM, 2023, pp. 427–430.

1.5 Terminology

To avoid confusion, below are the definitions of the terms commonly used in this thesis:

- A **problem** refers to a high-level proposition we aim at solving, such as the DFJSS problem.
- A **simulation** is a computer program that models the behaviour of the scheduling process. It replicates the essential features and functions of the real-world system, allowing users to study, analyse, or conduct experiments with it in a controlled virtual environment.
- An **instance** denotes a specific simulation conducted with a fixed random seed.
- A **scenario** represents a specific problem configuration to be solved, including instances generated with the same problem configuration, such as identical objectives and utilisation levels. An instance serves as an example of a scenario.
- A **case** is a subset of an instance, representing a segment of that instance. An instance can be divided into multiple cases.

1.6 Organisation of Thesis

The subsequent chapters of this thesis are structured as follows. Chapter 2 provides a comprehensive review of the essential background and related work. The primary contributions of this thesis are detailed in Chapters 3–6. Chapter 7 concludes the thesis. Figure 1.2 illustrates the outline of this thesis, featuring the main goals (denoted by ●) and employed techniques (denoted by ✓) for each chapter, along with inter-chapter connections. An overview of each chapter is shown as follows.

Chapter 2 presents the essential background of scheduling, the existing methods for JSS, and the basic concepts of GP and RL. This chapter also conducts a comprehensive review of existing literature on scheduling, with a particular focus on the involved techniques in this thesis.

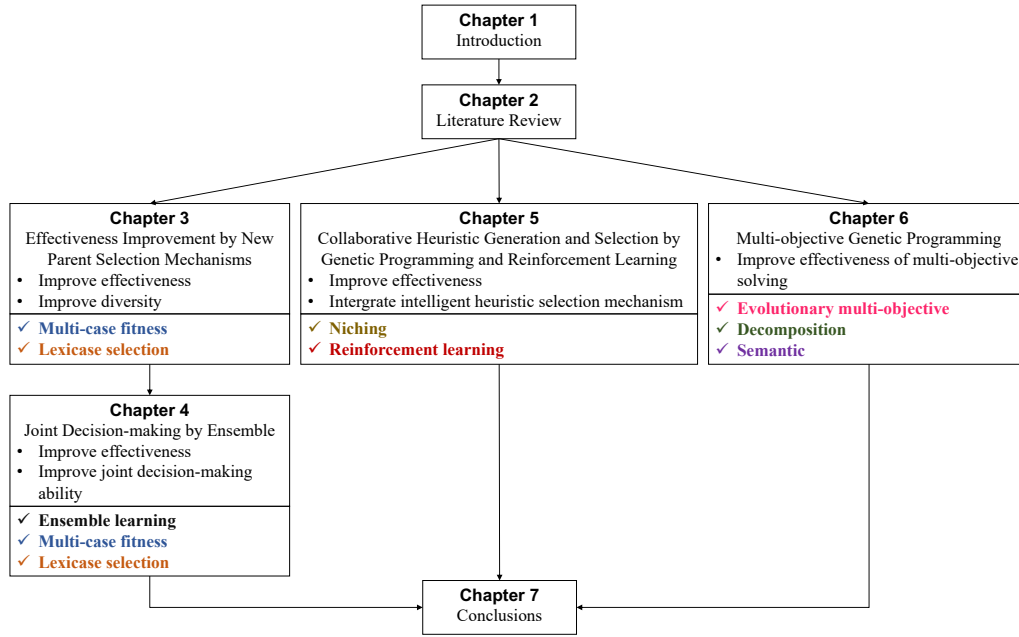


Figure 1.2: The outline of this thesis, including the main goals and involved techniques of each chapter, and the connection between chapters in this thesis.

Chapter 3 presents the proposed GP algorithm incorporating three newly developed parent selection mechanisms to enhance its efficacy in addressing the DFJSS problems. It begins with an overview of the overall framework of the proposed algorithms, followed by a detailed explanation of the key components. The chapter then introduces the experimental design, outlining how the experiments were conducted to validate the effectiveness of the proposed algorithms. Results and discussions follow, showing the efficacy of the proposed method in comparison to other methods. Further analyses are presented to provide insights into the reasons behind the proposed algorithm's effectiveness. Notably, the main techniques emphasised in this chapter are multi-case fitness and lexicase selection.

Chapter 4 introduces the proposed ensemble GP algorithm, designed

to enhance the effectiveness and joint decision-making ability of the evolved group of scheduling heuristics for DFJSS. It begins with an overview of the overall algorithmic framework, followed by a comprehensive explanation of key components, including population initialisation, individual evaluation, ensemble construction and selection, ensemble evaluation, and genetic operators. This chapter then provides an in-depth description of the experimental design, clarifying how the effectiveness of the proposed algorithm can be verified through experiments. Results and discussion follow, demonstrating the efficacy of the proposed method compared to other methods and highlighting the effectiveness of key components. Further analyses are presented along with this chapter to provide insights into the factors that contribute to the effectiveness of the proposed algorithm. Notably, the key techniques emphasised in this chapter include ensemble learning, multi-case fitness, and lexicase selection.

Chapter 5 presents the proposed two-stage framework designed for collaborative intelligent heuristic generation and selection through the integration of GP and RL to solve the DFJSS problems. This chapter begins with an overview of the algorithmic structure, followed by a comprehensive explanation of key components, including state features, niching GP, and deep RL. An in-depth description of the experimental design follows, showing how the effectiveness of the proposed algorithm can be verified experimentally. This is followed by results and discussion, demonstrating the effectiveness of the proposed method compared to other methods and highlighting the effectiveness of key components. Further analysis is provided in this chapter, which provides additional insight into the factors that contribute to the effectiveness of the proposed algorithm. Notably, the key techniques emphasised in this chapter are niching and RL.

Chapter 6 describes the proposed two multi-objective GP algorithms aimed at evolving a Pareto front of effective scheduling heuristics for addressing the multi-objective DFJSS problems. It first explores MOGP/D

and then presents the semantic NSGP-II algorithm. This chapter then provides a comprehensive overview of the experimental design, illustrating how the effectiveness of the proposed algorithm can be verified through experiments. The results and discussions illustrate the effectiveness of the proposed algorithm in comparison to other methods. Further analyses are conducted to provide an in-depth exploration of the factors that contribute to the effectiveness of the proposed algorithm. Notably, the main technique emphasised in this chapter is the evolutionary multi-objective, decomposition, and semantic.

Chapter 7 summarises the achieved objectives and the main conclusions of this thesis. Some discussions and future research directions are also presented in this chapter.

Chapter 2

Background

This chapter begins by introducing the details of the DFJSS problem, which is the focus of this thesis. Subsequently, fundamental concepts related to machine learning and evolutionary computation are outlined. Following this, the principles of GP, the distinction between heuristic and hyper-heuristic approaches, details on utilising scheduling heuristics for DFJSS, and the principles of RL are elucidated. Furthermore, this chapter provides a review of existing approaches for solving diverse JSS problems and summarises the research in this field. Additionally, related works, especially techniques related to the research objectives presented in this thesis are reviewed and summarised.

2.1 Dynamic Flexible Job Shop Scheduling

DFJSS [284] represents a sophisticated and challenging extension of the classical JSS problem. DFJSS introduces flexibility in terms of machine assignments, allowing a job operation to be processed on multiple machines. Unlike classical JSS, where only sequencing decisions are made (i.e., determining the order of operations on a fixed machine), DFJSS involves more intricate decisions. In DFJSS, decisions encompass not only selecting the operation to be processed by an idle machine but also choosing the specific

machine to process the ready operation [81]. Therefore, DFJSS involves two distinct decision points:

- **Routing decision point:** represents the situation when an operation becomes ready. At the routing decision point, a *routing rule* is required to select a machine for the ready operation.
- **Sequencing decision point:** represents the situation when a machine becomes idle. At the sequencing decision point, a *sequencing rule* is required to choose an operation from the waiting queue as the next task for processing.

Besides, DFJSS incorporates the dynamic nature of real-world manufacturing environments. Unlike the traditional flexible JSS, DFJSS relaxes the assumptions of having known information about jobs in advance. Instead, it introduces a more realistic scenario where job arrivals are dynamic, and their processing times remain unknown until they arrive. An extensive literature review has shown that the dynamic arrival of jobs is a prominent consideration in both practical scenarios and existing research studies [256, 257, 282, 284]. The detailed description of DFJSS is as follows.

In the shop floor, there are a set of machines $\mathcal{M} = \{M_1, M_2, \dots, M_m\}$. Jobs $\mathcal{J} = \{J_1, J_2, \dots, J_n\}$ that need to be processed arrive at the shop floor over time. The job information (e.g., operations, processing time of each operation, due date) is not known until it arrives at the shop floor. Each job J_i has an arrival time r_i , a weight w_i , a due date d_i , and consists of multiple operations $[O_{i,1}, O_{i,2}, \dots, O_{i,p_i}]$ that need to be processed in order. Each operation $O_{i,j}$ has a workload $\pi_{i,j}$, and can be processed by an optional machine in $\mathcal{M}_{i,j} \subseteq \mathcal{M}$. Each machine M_k has a unique processing rate γ_k . The processing time $t_{i,j,k}$ of operation $O_{i,j}$ on machine M_k is defined as $t_{i,j,k} = \pi_{i,j}/\gamma_k$. The machines are distributed, and there is a transport time τ_{k_1,k_2} [185] to transport a job between machines M_{k_1} and M_{k_2} . Following are common constraints, assumptions, and objectives associated with DFJSS.

Constraints and Assumptions

- An operation is only allowed to be processed after its preceding operation has been completed.
- The operation can only be processed by one of its optional machines.
- Recirculation of jobs is not allowed.
- Preemption is not allowed, meaning a machine cannot switch to process another operation until completing the current operation.

Objectives

Minimisation of:

- Max flowtime: $Fmax = \max \{C_1 - r_1, \dots, C_n - r_n\}$,
- Mean flowtime: $Fmean = \frac{\sum_{j=1}^n (C_j - r_j)}{n}$,
- Mean weighted flowtime: $WFmean = \frac{w_j \times \sum_{j=1}^n (C_j - r_j)}{n}$,
- Max tardiness: $Tmax = \max \{\max \{C_1 - d_1, 0\}, \dots, \max \{C_n - d_n, 0\}\}$,
- Mean tardiness: $Tmean = \frac{\sum_{j=1}^n (\max \{C_j - d_j, 0\})}{n}$,
- Mean weighted tardiness: $WTmean = \frac{w_j \times \sum_{j=1}^n (\max \{C_j - d_j, 0\})}{n}$.

where, C_i represents the completion time of job J_i .

It is noted that we do not consider all the objectives simultaneously in a many-objective case. Instead, we evaluate them individually as single objectives in various scenarios for single-objective DFJSS studies. In the case of multi-objective DFJSS studies, we examine pairs of objectives together for each scenario.

2.2 Basic Concepts and Approaches

2.2.1 Machine Learning

Machine learning [58] is a subset of artificial intelligence that focuses on the development of algorithms and models that allow computer systems to learn and make decisions. Machine learning algorithms can be broadly categorised into supervised learning, unsupervised learning, and reinforcement learning [110].

1. **Supervised learning** [48]: In supervised learning, the algorithm is trained on a labeled dataset, where the input data is paired with corresponding output labels. The model learns to map input features to the correct output based on labeled training data.
2. **Unsupervised learning** [17]: Unsupervised learning deals with unlabeled data, aiming to identify patterns, relationships, or structures within the data without explicit guidance. The algorithm explores the inherent structure of the data.
3. **Reinforcement learning** [111]: Reinforcement learning involves an agent interacting with an environment and learning to make decisions by receiving feedback in the form of rewards or penalties. The agent learns through trial and error to maximise cumulative rewards.

Machine learning involves training and test processes [110]. During the **training** phase, a model(s)/heuristic(s) is/are exposed to labeled or unlabeled data to learn patterns and relationships. The algorithm learns a model iteratively to minimise errors or improve performance. During the **test** phase, the trained model/heuristic is evaluated on new, unseen data to assess its **generalisation** capabilities. The goal is to ensure that the model/heuristic can make accurate predictions or decisions on data

it has not encountered during training. JSS falls into the category of reinforcement learning (sequential decision-making), where the scheduling decisions made by the system are based on feedback received from the environment.

2.2.2 Evolutionary Computation

Evolutionary computation is a family of optimisation algorithms inspired by the principles of Darwin's evolution [57]. Evolutionary computation mimics the process of natural evolution to search for optimal solutions to complex problems. The goal is to iteratively improve a population of candidate solutions through processes such as selection, mutation, and crossover, ultimately converging toward optimal or near-optimal solutions [12]. Evolutionary computation comprises two primary categories: *evolutionary algorithms* [11], such as genetic algorithms [96], GP [119], evolutionary programming [66] and evolution strategies [86], and *swarm intelligence* [117], such as particle swarm optimisation [68] and ant colony optimisation [149]. There are also other evolutionary computation algorithms, such as evolutionary multi-objective optimisation [198] and memetic computing [155], which also feature a population of candidate solutions but are not easily integrated into evolutionary algorithms or swarm intelligence. This thesis focuses on evolutionary algorithms.

An evolutionary algorithm begins with the creation of an initial population of individuals [5]. Each individual represents a potential solution to the given problem and is initially generated randomly. The fitness of each individual in the population is evaluated using a fitness function. The fitness function measures the effectiveness of an individual in addressing a given problem, offering a quantitative assessment of its quality by assigning fitness to the individual [57]. Fitness plays an important role in the selection process. Selection involves elitism selection and parent selection. Elitism selection ensures the best individuals from the current generation

are preserved and carried over to the next generation unchanged. These best individuals, often referred to as elites, are typically selected based on their fitness, with the top-performing individuals being retained. Elitism helps to maintain high-quality solutions in the population across generations, preventing the loss of promising individuals due to random variation or crossover and mutation operators. Parent selection is the process of choosing individuals from the current population to serve as parents for generating offspring in the next generation. Individuals are selected from the current population as parents based on their fitness. Individuals with better fitness are more likely to be selected, simulating the principle of “survival of the fittest” [12]. Various selection mechanisms, such as roulette wheel selection [184] or tournament selection [26], can be used. After parent selection, the breeding process generates offspring based on the selected parents by crossover, mutation, and reproduction. Crossover involves the exchange of genetic material between two parent individuals to create offspring [107]. In GP, crossover typically occurs at randomly selected points in the individuals’ genetic representation. This exchange of genetic material allows for the combination of beneficial traits from both parents, potentially producing offspring with improved fitness. Mutation involves making small, random changes to the genetic material of an individual. In GP, mutation can occur at various points in an individual’s genetic representation, such as changing a function or modifying a terminal. Mutation introduces genetic diversity into the population, helping to explore new regions of the search space and potentially discovering novel solutions. Reproduction is the process of selecting individuals from the current population to be carried over to the next generation unchanged. Reproduction ensures that successful individuals have the opportunity to pass on their genetic material to subsequent generations. These operators work together in the evolutionary process of GP to iteratively generate new generations of individuals. The algorithm continues evolving populations through multiple generations until a termination criterion is

met. Termination criteria can include a maximum number of generations, achieving a satisfactory solution, or reaching a specified level of convergence. Once the algorithm terminates, the best individual(s) in the final population are considered as the solution(s) to the problem.

2.2.3 Genetic Programming

GP is a powerful evolutionary algorithm that falls under the category of hyper-heuristic methods [120]. It is widely used for automated generation and improvement of heuristics to solve complex problems [112]. GP operates by evolving a population of heuristics to find optimal or near-optimal solutions. This section provides an overview of the traditional GP algorithm. Specifically, the representation and key processes of GP are introduced, including the initialisation, evaluation, selection, and evolution. The flowchart illustrating the steps of the traditional GP is presented in Figure 2.1.

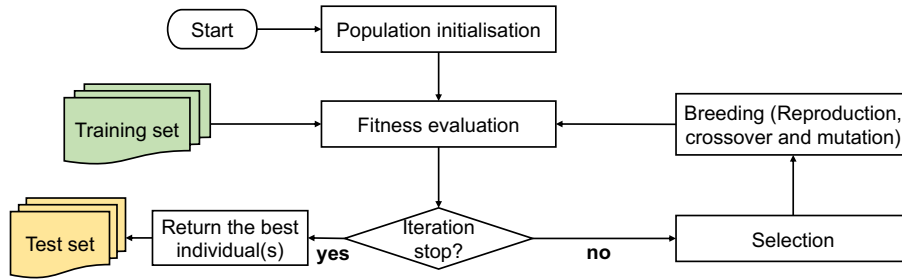


Figure 2.1: The flowchart of traditional GP.

Representation

The representation in GP defines the structure of individuals and is a critical aspect of the algorithm. Various representations exist, including tree-based structure [120], linear-based structure [102], grammar-based structure [16], and gene expression programming [64]. For DFJSS problems, the

commonly employed representation is the tree-based structure. In DFJSS, where two types of decision points need consideration (sequencing and routing), two potential tree-based representations can be used. One option involves a single-tree-based representation with two subpopulations, each dedicated to a specific type of decision point [160]. Another option is a multi-tree-based representation, where one tree is designated for routing rules, and the other is for sequencing rules [265]. In both single-tree-based and multi-tree-based representations, the tree is constructed by combining various functions and terminals. Functions typically denote mathematical or logical operations, such as $\{+, -, \times, /\}$, and are represented by internal nodes in the tree structure. Terminals represent scheduling information related to the system state, jobs, and machines on the shop floor and serve as the leaf nodes of the tree. In DFJSS, the multi-tree-based GP has demonstrated superiority over the single-tree-based GP with two subpopulations [265]. This is because the multi-tree-based GP allows for the concurrent evolution of routing and sequencing rules simultaneously, and can better capture interactions between these rules, leading to more effective solutions.

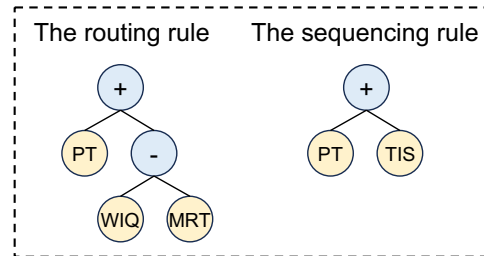


Figure 2.2: An example of multi-tree-based representation of a scheduling heuristic for DFJSS.

An illustrative example of the multi-tree-based representation for DFJSS is depicted in Figure 2.2, where PT, TIS, WIQ, and MRT are problem-specific features. PT denotes the processing time of operation. TIS represents the time in the system of a job. WIQ is the work in the

queue, representing the total processing time of operations in the queue. MRT means the machine ready time. If a smaller value represents a higher priority, the routing rule in Figure 2.2 prefers machines providing smaller processing time, smaller working in queue, and larger ready time. The sequencing rule in Figure 2.2 prefers operations with smaller processing time and smaller time in the system.

Initialisation

Initialisation is an important step in GP involving the creation of an initial population of candidate heuristics [33]. Individuals in the initial population are generated randomly. For tree-based representation, the size and structure of these trees are determined by parameters like maximum depth and generation methods. Three common tree generation methods are Full, Grow, and Ramped-half-and-half. The *Full* method ensures uniform depth across all leaf nodes, creating uniformly shaped trees with maximal depth [119]. The *Grow* method allows varied depths, providing more flexible structures [119]. Figure 2.3 illustrates examples of trees generated by the Full and Grow methods. Terminal nodes in the tree represent variables, constants, or input values related to the given problem, while function nodes represent operations or functions. The selection of functions and terminals during tree creation is randomised. The *Ramped-half-and-half* involves half the individuals generated using the Full method and the other half using the Grow method [119]. This method is widely adopted for initialisation in GP as it produces a more diverse population.

Evaluation

In GP, fitness evaluation is a crucial step that assesses the performance of individuals within the population based on their ability to solve a given problem [120]. The goal is to quantify how well each individual addresses the objectives of the problem. The fitness of an individual is commonly

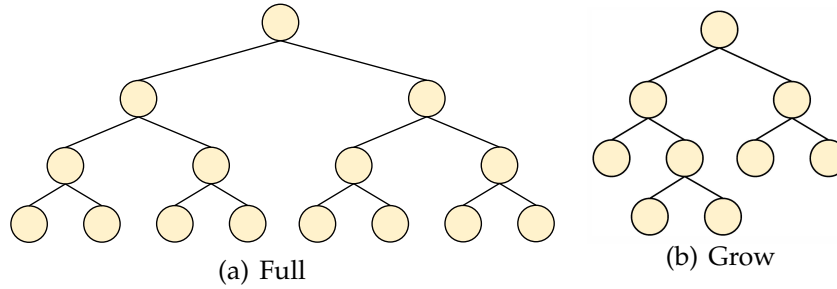


Figure 2.3: An example of individuals generated by Full and Grow methods.

evaluated through an objective function representing the specific criteria the algorithm aims to optimise. In DFJSS, this objective function can be the flowtime, tardiness, or other scheduling performance metrics [261]. To evaluate an individual's fitness, the GP individual is executed or simulated on training instances, producing a fitness value that quantifies how well the individual solves the problem. In scheduling with minimisation objectives, lower values are typically preferred (e.g., shorter flowtime, shorter tardiness). In scenarios involving multiple conflicting objectives (multi-objective optimisation), a multi-objective fitness function can be used [270]. This function combines diverse objectives into a single metric or evaluates each objective independently. To avoid the high computational cost associated with evaluating the fitness of each individual in the population, some studies propose using surrogate models [76]. These models serve as efficient approximations of the true fitness function [263].

Fitness values play a crucial role in the selection process, influencing the probability of individuals being chosen as parents for breeding. For problems with minimisation objectives, smaller fitness values increase the probability of an individual being selected, promoting the evolution of better solutions. Overall, fitness evaluation is important to the success of GP, as it guides the evolution of the population toward better solutions over generations. The effectiveness of the algorithm relies on the accurate representation of problem-specific objectives in the fitness function.

Selection

Selection is essential in GP that determines the direction of evolution [90]. Selection can be divided into elite selection and parent selection. *Elite selection* involves preserving a certain percentage of the best individuals from the current generation to be directly carried over to the next generation. This helps maintain high-performing individuals in the population. *Parent selection* determines which individuals from the current population will be chosen to serve as parents for creating the next generation [62]. The choice of parents significantly influences the diversity and quality of the population.

Individuals with better fitness values are more likely to be selected as parents. This approach emphasises the principle of “survival of the fittest” and aims to propagate genetic material from individuals that have performed well in solving the problem. Tournament selection and roulette wheel selection are two parent selection examples commonly used in GP.

Tournament selection involves randomly selecting a subset of individuals from the population and choosing the one with the highest fitness as a parent [62]. The number of sampled individuals is set manually in advance. *Roulette wheel selection* allocates a probability of selection to each individual in proportion to their fitness [228]. Individuals with better fitness have larger “slices” of the roulette wheel, increasing their chances of being chosen. The choice of parent selection operator depends on the specific characteristics of the problem, the desired balance between exploration and exploitation.

Breeding

The breeding process in GP involves the creation of new individuals (offspring) by combining genetic material from selected parents [120]. The primary genetic operators used in GP are crossover, mutation, and reproduction.

Crossover involves exchanging genetic material between two parents to generate one or more offspring. Crossover facilitates the recombination of building blocks from different parents, allowing the offspring to inherit good combinations of genetic material. Different crossover methods exist, such as subtree crossover and one-point crossover. For example, subtree crossover selects a subtree from one parent and replaces a randomly chosen subtree in the other parent [265]. An example of the crossover process in GP for DFJSS can be seen in Figure 2.4.

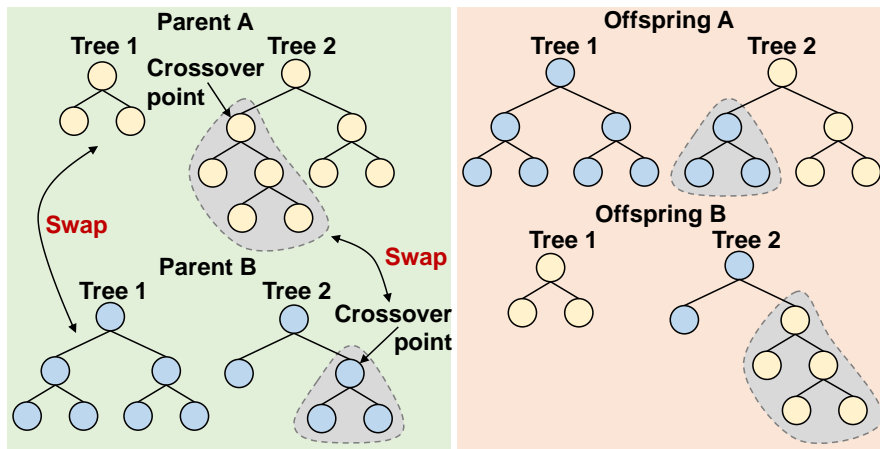


Figure 2.4: The process of crossover.

Mutation is a genetic operator that introduces small random changes to an individual's genetic material [120]. Mutation introduces genetic diversity, helping the population explore new regions of the solution space that might lead to improved solutions. Mutation can take various forms, such as changing a function or terminal in a subtree, adding or deleting nodes. Mutation serves as an exploitation mechanism, allowing the algorithm to discover novel solutions and prevent premature convergence. An example of the mutation process in GP for DFJSS can be seen in Figure 2.5.

Reproduction is a genetic operator that allows certain individuals (not necessarily the best ones) to be directly copied into the next generation without modification [119]. This process preserves high-quality genetic

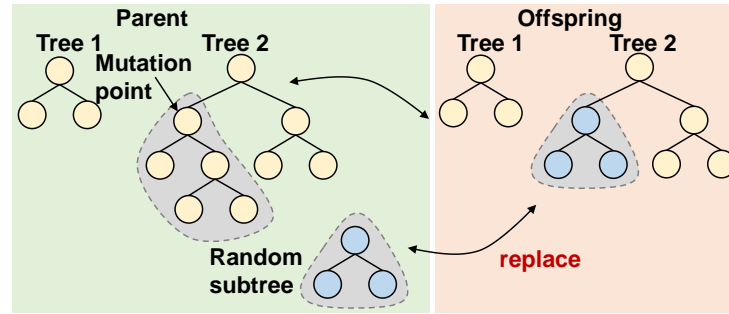


Figure 2.5: The process of mutation.

material, ensuring that superior individuals from the current generation persist in contributing to the population. By retaining high-quality individuals, reproduction helps maintain consistency and stability within the population.

These genetic operators often work simultaneously. Typically, a combination of crossover, mutation, and reproduction is applied to generate offspring in a new generation [119]. Moreover, crossover typically has the highest probability of occurrence, followed by mutation, and lastly, reproduction in GP for DFJSS [267].

2.2.4 Heuristic and Hyper-heuristic

A **heuristic** approach refers to a problem-solving strategy or technique for finding approximate solutions to problems [14]. Heuristics are frequently employed in dealing with complex or computationally demanding problems. While heuristic methods do not ensure optimal solutions, they offer rapid, practical, and often satisfactory results [263]. Simple rules, such as scheduling heuristics that prioritise candidate machines or operations, are considered heuristics in specific domains, like scheduling problems [263].

A **hyper-heuristic** [32] is a form of heuristic learning approach that distinguishes itself from traditional heuristics by operating on a set of heuristics rather than directly generating solutions in the problem's solu-

tion space. Hyper-heuristic approaches can be divided into two categories, which are *heuristic selection* [25] and *heuristic generation* [33]. The former one selects from existing heuristics for different situations, the latter one generates new high-level heuristics from existing low-level heuristics.

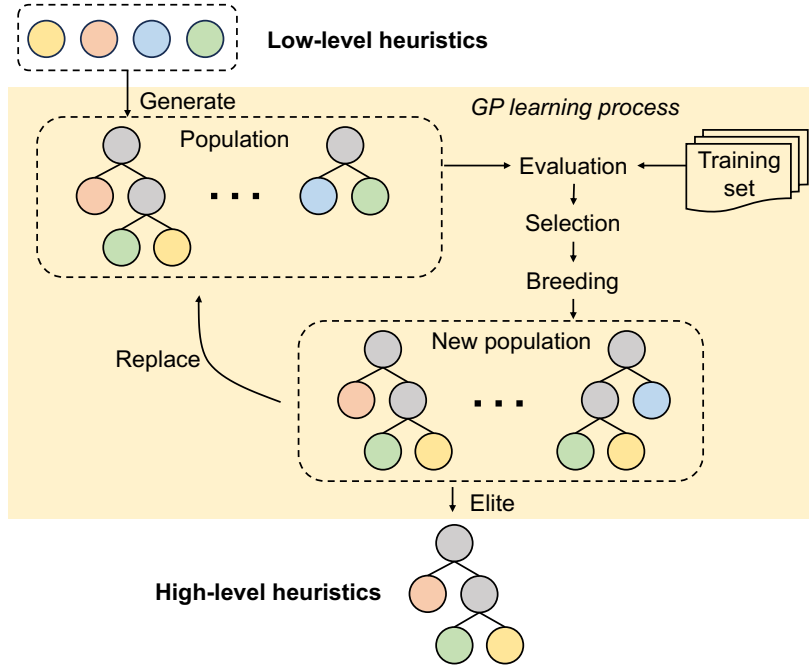


Figure 2.6: An example of generating high-level heuristic from simple low-level heuristics with GP.

The relationship between heuristics and hyper-heuristics lies in their complementary roles in the optimisation process. Heuristics serve as the fundamental building blocks or strategies for solving specific types of problems. They are the low-level, problem-specific approaches that can be effective in certain situations. Hyper-heuristics operate at a higher level and coordinate the use of heuristics [33]. They aim to optimise the selection or generation of heuristics based on the characteristics of the given problem. While heuristics are often designed for specific problem types, hyper-heuristics offer adaptability. In DFJSS, GP serves as a typical hyper-

heuristic approach for generating heuristics. Take GP as an example, Figure 2.6 shows how to generate high-level heuristics from low-level heuristics by GP. The four circles, each in a distinct color, represent different low-level heuristics, while the circles in gray color denote functions such as $\{+, -, \times, /\}$. The GP learning process is illustrated within the yellow box. Initially, a population of high-level heuristics is randomly generated. Subsequently, these high-level heuristics are iteratively improved through processes of evaluation, selection, and breeding across generations. Ultimately, the best high-level heuristic(s) is the output of the GP learning process.

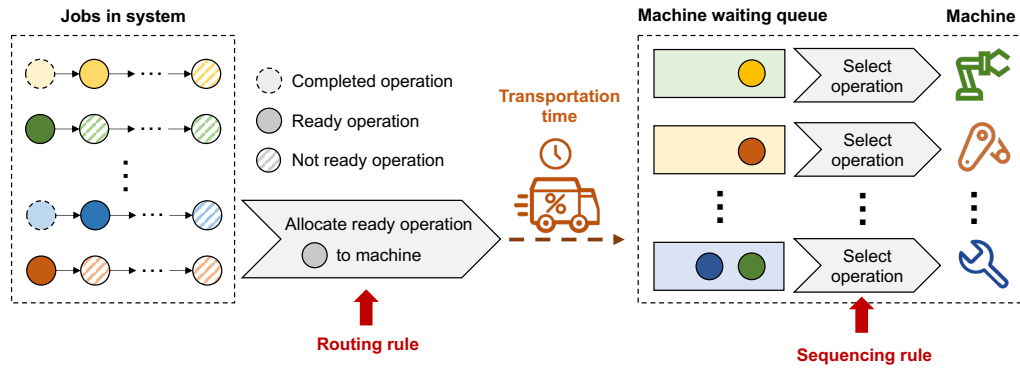


Figure 2.7: An example of DFJSS decision process.

2.2.5 Scheduling Heuristics

A scheduling heuristic is a priority function used to calculate the priority of the candidate machines or operations based on the system state and the state of the candidate machines and operations. In JSS and dynamic JSS, a scheduling heuristic only includes one sequencing rule, which is used to select operations when a machine is idle. In flexible JSS and DFJSS, a scheduling heuristic consists of two rules: a *routing rule* and a *sequencing rule*, which are used when the decision point arrives. An example of DFJSS and how to use the scheduling heuristic with a routing rule and a

sequencing rule is shown in Figure 2.7. As we can see, once an operation becomes ready (routing decision point), the routing rule will select a machine to process this operation. Once a machine becomes idle (sequencing decision point), the sequencing rule will select an operation as the next task for this machine. The most commonly used scheduling heuristics include WIQ (the works in the queue), PT (the processing time of operation), and SL (the slack) [258, 265]. Scheduling heuristic is an efficient method to make fast decisions at decision points and can cope well with dynamic events.

2.2.6 Reinforcement Learning

RL [55] is a subfield of machine learning and is normally used as a hyper-heuristic method. RL is inspired by the way animals learn through trial and error by interacting with their surroundings. Basic RL is modeled as a Markov decision process [183] which can be defined by a tuple (S, A, ϕ, R, γ) , where:

- S is the set of states,
- A is the set of actions,
- ϕ is the state transition probability function: $\phi(s_{t+1}|s_t, a_t)$,
- R is the reward function: $R(s_t, a_t, s_{t+1})$,
- γ is the discount factor ($0 \leq \gamma \leq 1$).

An agent interacts with the environment in discrete time steps. At each time step t , the agent observes the current state s_t , selects an action a_t from the set of possible actions A , and receives a reward r_t according to the reward function. The environment transitions to a new state s_{t+1} based on the state transition probability. The agent's goal is to learn a policy $\pi(a|s)$, a mapping from states to actions, that maximises the expected cumulative reward [55]. Figure 2.8 gives the general framework of RL.

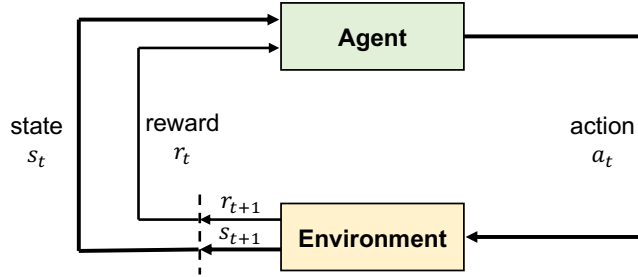


Figure 2.8: The general framework of RL [153].

RL encompasses various representations for the policy (heuristic), which are not only influenced by the learning algorithms but also closely tied to the nature of the specific problem being addressed. Different representations exist in RL, including feedforward deep neural networks [56, 131], recurrent neural networks [151], and graph neural networks [170, 250]. Each of these representations requires problem-specific features as input. RL utilises learning algorithms to guide the agent in updating its policy based on observed experiences. Common learning algorithms include Q-learning [186], deep Q-networks (DQN) [129], and double DQN [216]. Among these representations and learning algorithms, the deep neural network and double DQN are particularly popular for solving DFJSS problems. Taking RL with the deep neural network and double DQN as an example, the details about key components in RL, including representation, initialisation, action selection and evaluation, and policy updating, are introduced as follows.

Representation

The double DQN introduces the concept of using two separate Q-networks: the online Q-network (Q) and the target Q-network (Q'). The online Q-network (Q) is used for action selection, representing the policy that an agent follows to interact with its environment. The policy is often represented by a deep neural network that takes the state of the environ-

ment as input and outputs Q-values for each possible action.

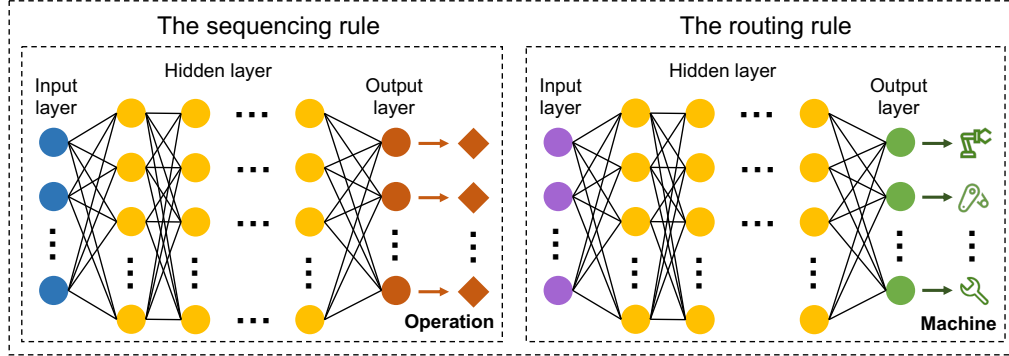


Figure 2.9: An example of end-to-end neural network-based representation of a scheduling heuristic for DFJSS.

An illustrative example of the feedforward deep neural network-based representation for DFJSS is depicted in Figure 2.9, which includes the arrangement of neurons (nodes) and connections (edges) between them. These layers can be broadly categorised into three types: the input layer, hidden layers, and the output layer. To be specific, the input layer represents the features or input variables related to the DFJSS scheduling system. Each node in the input layer corresponds to a specific feature of the input data (e.g., the scheduling state in DFJSS). The number of nodes in the input layer is determined by the dimensionality of the input data. Between the input and output layers, there can be one or more hidden layers. Each node in a hidden layer represents a learned feature or abstraction based on the input data. The output layer produces the final results of the neural network's computation. Each node in the output layer represents a numerical value, representing the priority of each action. For direct (end-to-end) RL, the actions directly represent the machines or operations, as shown in Figure 2.9. For indirect RL, the actions correspond to the scheduling heuristics. Once the scheduling heuristic is determined, it subsequently makes the final decision regarding the machines or operations to be scheduled. The number of nodes in the output layer is determined

by the desired output of the network. Each connection between nodes has an associated weight, representing the strength of the connection. Information flows from the input layer, through the hidden layers, and finally to the output layer through these weighted connections. To be noticed, the number of hidden layers and nodes in each input/hidden/output layer is a design choice and can vary based on the complexity of the problem and can vary for sequencing and routing rules.

Initialisation

The RL process starts with initialising the online Q-network (Q), the target Q-network (Q'), and the replay memory. Common Q-network initialisation methods include random initialisation or using pre-trained weights from a prior related task. Proper initialisation is crucial to ensure a good starting point for learning. Normally, random initialisation is employed. The replay memory is initialised empty and is used to store experience (s_t, a_t, r_t, s_{t+1}) for training the online Q-network.

Action Selection and Evaluation

The action selection process involves choosing an action based on the state s_t of the environment at time step t . In double DQN, the agent estimates Q-values for each action using its Q-networks. The action a_t is then selected based on a certain exploration-exploitation strategy, such as epsilon-greedy [216], where the agent chooses the action with the highest estimated Q-value with high probability, but occasionally explores other actions. After selecting an action, the agent executes it in the environment and receives a reward r_t and the next state s_{t+1} . The Q-value update rule in DQN is given by the Bellman equation [166] and is represented as Eq. (2.1)

$$Q(s_t, a_t) = (1 - \alpha_t)Q(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_{a'} Q(s_{t+1}, a') \right) \quad (2.1)$$

where $Q(s_t, a_t)$ is the Q-value for state s_t and action a_t . α_t is the learning rate. r_t is the observed reward after taking action a_t in state s_t . γ is the discount factor. s_{t+1} is the next state and $\max_{a'} Q(s_{t+1}, a')$ is the maximum Q-value for the next state s_{t+1} .

Traditional DQN methods have an overestimation bias issue. Double DQN helps mitigate the overestimation bias present in traditional DQN methods, leading to more stable and accurate value estimates [216]. The use of two Q-networks, with one for action selection and one for value estimation, enhances the learning process and contributes to better policy updates. The Q-value of the selected action in double DQN is updated based on the observed reward and the estimated Q-value of the next state, which is represented as Eq. (2.2).

$$Q(s_t, a_t) = (1 - \alpha_t)Q(s_t, a_t) + \alpha_t (r_t + \gamma Q'(s_{t+1}, \operatorname{argmax}_{a'} Q(s_{t+1}, a')))) \quad (2.2)$$

where α is the learning rate. $Q(s_t, a_t)$ is the Q-value for state s_t and action a_t in the online Q-network, $Q'(s_{t+1}, \operatorname{argmax}_{a'} Q(s_{t+1}, a'))$ is the Q-value for the next state s_{t+1} using the target Q-network, and $\operatorname{argmax}_{a'} Q(s_{t+1}, a')$ selects the action that maximises the Q-value in the online Q-network.

The goal is to maximise the cumulative reward as Eq (2.3). A well-designed reward function is crucial for training a high-quality policy for RL.

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \quad (2.3)$$

where γ is the discount factor, and r_{t+k+1} is the reward at time $t + k + 1$.

Policy Updating

The policy updating process involves adjusting the parameters of the Q-networks to improve the accuracy of Q-value predictions. In double DQN, this is done through a combination of minimising the temporal difference error and updating the Q-networks' weights using optimisation tech-

niques like stochastic gradient descent [216]. The online Q-network (Q) is used for action selection and is updated more frequently. The target Q-network (Q') is periodically updated to match the online Q-network's parameters. This helps stabilise the learning process and reduce overestimation bias.

The RL with double DQN involves a continuous cycle of action selection and policy updating. The learning process aims to refine the policy, enabling the agent to be more proficient at making decisions in complex and dynamic scenarios.

2.3 Job Shop Scheduling Approaches

This section provides a comprehensive investigation of various approaches for solving JSS problems, including classical approaches and their variations. The main approaches are categorised into three classes, which are *exact approaches*, *heuristic approaches*, and *hyper-heuristic approaches*. The common approaches used for JSS and their classifications are shown in Figure 2.10.

2.3.1 Exact Approaches

Exact approaches for solving JSS problems involve algorithms and techniques that guarantee the identification of an optimal solution within a reasonable amount of time. These methods are particularly effective for smaller problem instances, where the computational complexity remains manageable. Some key exact methods used in JSS include mathematical programming [221], integer linear programming [133], branch-and-bound [281], and dynamic programming [40]. *Mathematical programming* involves formulating a mathematical model that represents the JSS problem, typically in the form of objective functions and constraints [221]. The objective function quantifies the performance measure to be optimised, such

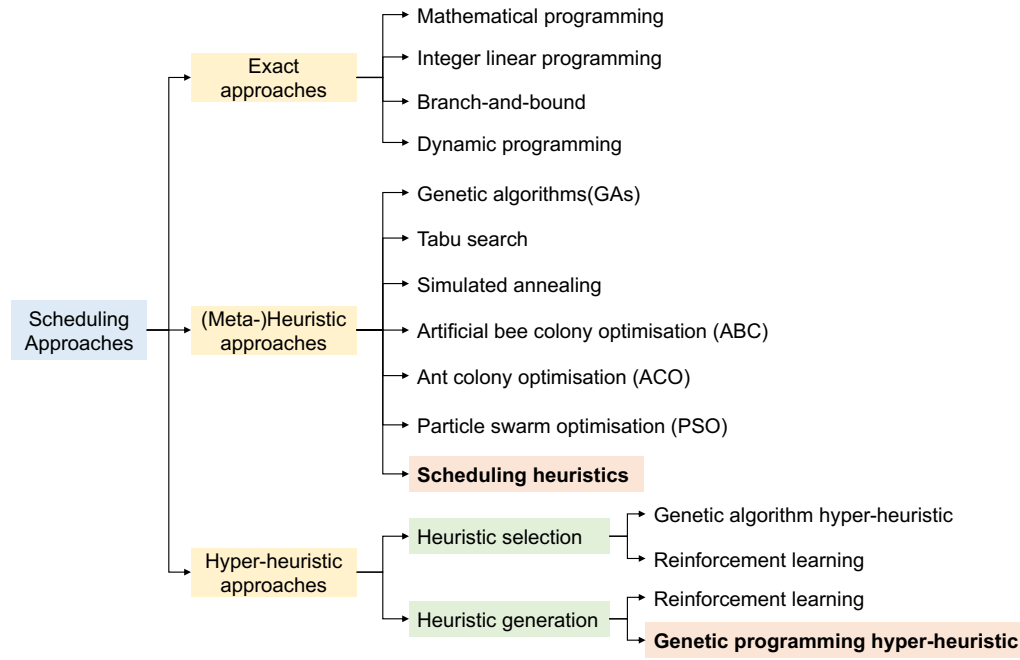


Figure 2.10: The common approaches used for JSS and their classifications.

as minimising makespan or total completion time, while constraints capture the limitations and requirements of the scheduling problem, such as machine capacities, precedence relationships between operations, and resource constraints. *Integer linear programming* is a specific type of mathematical programming where decision variables are restricted to integer values [133]. In the context of JSS, integer linear programming is often used to formulate scheduling problems as optimisation models with binary or integer decision variables representing the assignment of operations to time periods or machines. *Branch-and-bound* is a widely used optimisation technique that explores the solution space by dividing it into subproblems and bounding the solution space based on certain criteria. It efficiently prunes branches of the solution tree that cannot lead to an optimal solution, reducing the overall search space. Some studies of using branch-and-bound for JSS can refer to [4, 6]. Branch-and-bound with time-indexed formulation is a variation of branch-and-bound that formu-

lates the problem in a time-indexed representation [9], allowing for a more efficient exploration of the solution space. It exploits the temporal aspects of the problem to enhance pruning and reduce the search space. Some other variations of branch-and-bound methods that were proposed and successfully applied to many small-scale scheduling problems can be seen in [182, 193]. *Dynamic programming* [40] is often used for specific cases, such as in the context of parallel machine scheduling. It breaks down the scheduling problem into smaller subproblems and utilises optimal solutions to these subproblems to construct an optimal solution for the overall problem. Some studies of using dynamic programming for JSS can refer to [40, 83, 165, 204].

Although exact methods guarantee optimal solutions, they often become computationally intractable for larger problem instances due to their inherent complexity [274]. Additionally, exact methods struggle to adapt to dynamically changing parameters.

2.3.2 (Meta-)Heuristic Approaches

(Meta-)Heuristic approaches play a key role in addressing the JSS problems, particularly for large and complex instances where exact approaches are computationally infeasible. Unlike exact approaches, (meta-)heuristic approaches do not guarantee to find an optimal solution. Instead, they aim to find good enough solutions. (Meta-)Heuristic approaches can be categorised into two groups: iterative improvement heuristic approaches and constructive heuristic approaches.

Common *iterative improvement heuristic* approaches for JSS include genetic algorithm [82, 141], Tabu search [190, 219], simulated annealing [8], artificial bee colony optimisation [46, 277], ant colony optimisation [89, 99], and particle swarm optimisation [130, 132]. Genetic algorithm is inspired by the process of natural selection [233]. The genetic algorithm explores the solution space efficiently, leveraging the principles of survival of the

fittest to converge towards good-quality solutions [179]. Solutions to the scheduling problem are represented as chromosomes, and genetic operations such as crossover and mutation are applied to evolve a population of solutions. Tabu search [79] starts from an initial feasible solution and explores the neighborhood of the current solution iteratively. To avoid getting trapped in a local optimum, a flexible “memory” technique (Tabu list) is used in the Tabu search to record and select the optimisation process that has been performed and to guide the search direction for the next step. In [190], Tabu search was proposed for flexible JSS, which is composed of two parts. One part is used to identify the optimal sequence of operations, while the other focuses on determining the optimal choice of machines. This study shows that the Tabu search achieves better performance when compared to the branch-and-bound method. Furthermore, in [63], an innovative circular algorithm based on Tabu Search was introduced. This algorithm is designed to enhance both the exploration and exploitation aspects of the program, ultimately leading to improved solutions. Simulated annealing is a probabilistic optimisation algorithm that mimics the annealing process in metallurgy. The algorithm accepts moves that lead to both better and worse solutions, allowing it to escape local optima [22]. The ant colony optimisation, bee colony optimisation, and particle swarm optimisation are nature-inspired optimisation algorithms that found widespread application in solving diverse scheduling problems, as evidenced by studies such as [46, 59, 67, 100, 220]. Each algorithm leverages the principles of collective intelligence observed in social insects or particles to guide the search for high-quality schedules.

Overall, the iterative improvement heuristic approaches can find good solutions and can handle large-scale problems well. However, they are not suitable for dynamic JSS, since the rescheduling process inherent in iterative improvement heuristic approaches is still time-consuming, making it inefficient for reacting to dynamic events.

Constructive heuristic approaches (e.g. scheduling heuristics) offer real-

time schedule generation, making them suitable for dynamic JSS problems. These approaches assign priorities to operations or machines at decision points based on the system state [34]. Then, operations and machines are dispatched based on their priorities, with the highest-priority operation or machine being selected first. Common scheduling heuristics include First-In-First-Out, Shortest Processing Time First, Longest Processing Time First, and Most Operations Remaining First [196]. The choice of a scheduling heuristic depends on the specific practical problems or customer requirements. For instance, if minimising the mean flowtime of all jobs is the objective, the Shortest Processing Time First might be the preferred choice. It's worth noting that a single scheduling heuristic might not perform well across various scenarios, and a number of different heuristics are often required. However, designing effective scheduling heuristics demands a lot of domain knowledge and can be a time-consuming process. Moreover, it is challenging to manually design scheduling heuristics that can perform well on multiple objectives simultaneously.

2.3.3 Hyper-heuristic Approaches

Hyper-heuristic approaches [30], including both heuristic selection and heuristic generation, represent advanced approaches in solving hard computational search problems. These approaches operate at a high level, automating the process of selecting or generating heuristics to optimise scheduling performance. Both heuristic selection and generation are particularly valuable in dynamic environments where the optimal heuristic might change over time and are effective in addressing complex problem structures where predefined heuristics might not be sufficiently adaptive. In this case, hyper-heuristic approaches are quite suitable for solving DFJSS problems.

Heuristic selection involves selecting from a predefined set of existing heuristics based on their performance in specific situations. The selection

mechanism relies on historical information or problem-specific features to identify the most suitable heuristic for the current situation. This adaptability allows the hyper-heuristic to dynamically choose the most effective heuristic for different instances. Genetic algorithm (GA) hyper-heuristic is an example of heuristic selection. In [61, 188], GA hyper-heuristic is used to solve the JSS problems. To be specific, GA hyper-heuristic navigates the space of sequences of heuristic choices. In a real-world scheduling challenge presented in [87], the problem involved scheduling the collection and delivery of live chickens from farms in Scotland and northern England to processing plants, considering specific order requirements. The proposed GA hyper-heuristic in this scenario incorporates two GAs. One GA is dedicated to selecting heuristic rules for managing order allocation, while the other GA focuses on scheduling the arrival of deliveries. The method has achieved good results in practical applications. RL can also serve as a heuristic selection method for JSS. In this context, RL is often applied by considering manual scheduling heuristics as actions. The process entails utilising RL to dynamically select appropriate actions when encountering decision points. RL becomes particularly valuable when the number of candidate machines or operations is not fixed at different decision points, posing a challenge in defining a predetermined action space. [36, 84, 136, 139] are some examples of using RL for making heuristic selections to solve JSS problems.

Heuristic generation goes beyond using predefined heuristics by creating new heuristics or combining existing ones to adapt to the given problem. It dynamically learns or evolves heuristics during the optimisation/learning process. RL can serve as a heuristic generation method for JSS. However, the application of heuristic generation RL is often limited to static and non-flexible environments. Dynamic JSS problems, where the number of candidate machines or operations can change on the shop floor, pose a significant challenge for heuristic generation RL. GP is a popular heuristic generation method, which evolves a population of heuris-

tics over multiple generations. GP has been widely used in automatically evolving scheduling heuristics for solving JSS problems. In [163], the existing studies about using GP and its variants for solving JSS problems are summarised to provide an overall picture. Each of these variants of GP is able to evolve scheduling heuristics that are superior to most of those manually designed ones by human experts in the literature. A more comprehensive review of using GP for automatically learning scheduling heuristics to solve JSS problems is detailed in Section 2.4.

Overall, hyper-heuristic approaches, whether through heuristic selection or generation, represent a powerful paradigm for addressing JSS problems. These approaches leverage adaptability and computational intelligence to dynamically choose or create heuristics, ultimately enhancing scheduling performance across diverse instances of the JSS problems. Here, we have provided a general overview of existing approaches, encompassing exact approaches, heuristic approaches, and hyper-heuristic approaches, for various types of JSS problems. Next, we will concentrate on a comprehensive review of the relevant literature that aligns with our contributions.

2.4 GP for Evolving Scheduling Heuristics

In recent years, GP has been widely used to solve different kinds of JSS problems. In [157], a unified framework for the automated design of scheduling heuristics with GP was developed and the overall picture of how GP can be applied for evolving scheduling heuristics was provided. In this section, we will introduce different kinds of GP methods designed for different kinds of JSS problems. In this thesis, the JSS problems are classified according to environmental conditions. They are classified into static scheduling problems, including static JSS and static flexible JSS problems, and dynamic scheduling problems, including dynamic JSS and DFJSS problems.

2.4.1 Static Scheduling Problems

While GP is commonly employed for learning scheduling heuristics in dynamic JSS, its applicability extends to static JSS as well. In the domain of using GP for static JSS, it can be treated either as a learning approach or an optimisation approach [262]. When GP is used as a learning approach, GP operates similarly to its application in dynamic JSS. It learns scheduling heuristics based on training instances, and the learned heuristics are subsequently tested on an unseen test set to assess performance. When GP is used as an optimisation approach, GP can be utilised directly to obtain solutions or schedules for given problems, skipping the training and testing phases usually associated with learning approaches. Following are some examples illustrating the application of GP in JSS and flexibleJSS.

Static Job Shop Scheduling: In the early stage, GP is applied to address straightforward JSS problems [150]. The experimental results revealed that GP could evolve effective scheduling heuristics for solving such problems. Building upon this foundation, Nguyen et al. conducted a computational study of representations in GP for JSS in [159]. Their findings conclude that integrating system and machine attributes in the representation significantly enhances the quality of evolved scheduling heuristics. Then, to further improve the performance, Nguyen et al. proposed a new kind of scheduling heuristic called iterative scheduling heuristic that can iteratively refine schedules by leveraging the information recorded from previous or existing schedules. The work presented in [162] proposed two selection strategies to help surrogate-assisted GP in evolving scheduling heuristics for JSS. Both strategies are used to select diverse individuals for the next population. One is to enhance population diversity by selecting individuals with different behaviours (phenotypes), and the other one is to choose individuals with different genotypes. In [143], an efficient feature selection strategy was proposed for GP to solve the JSS problems. To be specific, a niching-based search framework was developed to extract a diverse set of good scheduling heuristics, and a surrogate model

was used to reduce the complexity of fitness evaluation. In addressing a specific JSS problem with lot-sizing issues, a cooperative coevolution GP was proposed in [248]. This method obtains effective performance, emphasising the efficacy of GP in solving specific challenges within the JSS problem domain.

In [113], the focus is on addressing the multi-objective JSS problems using a novel GP method assisted by the island model. The experimental results reveal that this innovative GP method, in conjunction with the island model, exhibits superior performance compared to classical multi-objective optimisation methods, including NSGAII and SPEA2. The paper considers three objectives. In a related exploration, the many-objective JSS problem is studied, which involves considering four or more objectives [140]. To tackle this complex problem, the study proposes a new hybridised algorithm that combines GP and NSGAIII, showcasing a comprehensive approach to address the challenges posed by many objectives in the JSS domain.

Static Flexible Job Shop Scheduling: In [95], GP is employed to evolve scheduling heuristics for solving the flexible JSS problem. The evolved scheduling heuristics demonstrate superiority over manually designed scheduling heuristics. In a more specialised context, Zhu et al. [286] proposed the application of GP to address a specific flexible JSS problem incorporating multi-process planning, showcasing the effectiveness of GP in solving specific challenges within the flexible JSS problem domain. In [28], an investigation of both single-tree and multi-tree representations in GP for solving the flexible JSS problem was conducted. For the single-tree representation, the work involves learning sequencing and routing rules in two parallel GP methods, while the GP with multi-tree representation simultaneously learns scheduling heuristics with sequencing and routing rules. The effectiveness of both representations was validated on benchmark flexible JSS instances against manually designed scheduling heuristics. Furthermore, this study extends its evaluation to dynamic and real-

world flexible JSS instances, verifying the effectiveness of the proposed methods.

In [206], the multi-objective flexible JSS problem was studied with three objectives taken into consideration. However, these objectives are linearly combined into a single objective, transforming the original multi-objective flexible JSS problem into a single-objective flexible JSS problem. In [285], a multi-agent-based GP was proposed for solving the flexible JSS problem, with a focus on a real-world application of GP in an Aero-Engine blade manufacturing plant.

Based on the literature review, there are limited studies about using GP for solving the JSS and flexible JSS problems. Normally, the dynamic is a more important issue, bringing scheduling scenarios closer to real-world applications and requiring greater attention.

2.4.2 Dynamic Scheduling Problems

Differing from static JSS, dynamic JSS involves incomplete advanced knowledge of job or machine information. Generally, GP is employed as a learning algorithm, with the expectation that the learned scheduling heuristics from the training process exhibit strong generalisation capabilities to effectively handle a range of unseen instances. Following are some examples illustrating the application of GP in dynamic JSS and DFJSS.

Dynamic Job Shop Scheduling: In the field of dynamic JSS, GP is applied to evolve effective scheduling heuristics for dynamic JSS, considering various dynamic events. Example studies such as [60, 73, 114, 237] focus on GP for dynamic JSS, particularly in the context of new job dynamic arrival events. These studies validate the effectiveness of GP against manually designed scheduling heuristics. Enhanced GP methods, integrating various machine learning techniques such as feature selection [195], multi-task learning [102], ensemble, niching [171], and surrogate, have been studied to further improve the effectiveness of GP in dynamic JSS. No-

table works, including [175, 194, 247, 253], explore these techniques and demonstrate improved performance over traditional approaches. Some studies extend their focus to consider both new job dynamic arrival events and machine breakdown events. For instance, in [174], simultaneous consideration of dynamic job arrivals and machine breakdowns is addressed, with the introduction of new machine breakdown terminals designed for GP. In the above studies, a single-tree representation is employed to learn scheduling heuristics with a single rule. Nevertheless, in specific scenarios, employing two rules can yield superior results. Differing from the sequencing rule and routing rule typically utilised in scheduling heuristics for flexible JSS, these two rules consist of a general sequencing rule and a specialised rule tailored for a specific goal. For example, in [173], a niched GP method combined with multitasking was proposed to evolve two rules simultaneously, effectively handling various machine breakdown scenarios. This approach generates sequencing rules effective across the entire dynamic JSS problem and specialist rules specifically tailored for handling machine breakdown issues. In [158], scheduling heuristics with a sequencing rule and a due-date assignment rule are evolved for dynamic JSS with multiple objectives. The due-date assignment rule is evolved specifically for optimising the tardiness objective. In addition to the effectiveness, certain studies focus on the interpretability of evolved scheduling heuristics by GP for dynamic JSS [103, 144].

Moreover, several studies concentrate on the application of GP to address multi-objective dynamic JSS. For example, in [160], four novel multi-objective GP methods for evolving scheduling heuristics in multi-objective dynamic JSS were proposed. These methods are formulated by integrating GP with classical multi-objective algorithms, including NSGAI and SPEA2. In [116] a novel GP for evolving sampling heuristics for multi-objective dynamic JSS was developed. Specifically, during the evolutionary process, sampling heuristics are employed to discard poor instances in favor of good instances, thereby enhancing the Pareto front. More studies

of GP for multi-objective dynamic JSS can refer to [1, 158, 161, 164, 243].

Dynamic Flexible Job Shop Scheduling: DFJSS is more complex compared to other types of JSS due to complicated interactions between sequencing and routing decisions in a dynamic environment. DFJSS has been attracting more and more attention from researchers due to its practical value. In addressing the challenge of managing multiple sequencing and routing scheduling decisions in a job shop, two typical GP methods are proposed, which are the cooperative coevolution GP method and the multi-tree GP method. In the cooperative coevolution GP approach, the sequencing rule and routing rule are evolved in two distinct subpopulations [245]. On the other hand, the multi-tree GP method evolves both sequencing and routing rules simultaneously within a single population, with each individual comprising two trees [265]. Research indicates that both of these methods can achieve superior performance compared to manually designed scheduling heuristics. On top of that, certain improved cooperative coevolution GP methods and multi-tree GP methods are developed by combining them with one or multiple machine learning techniques, such as feature selection, surrogate, multi-task learning, knowledge transfer, and ensemble for DFJSS [262]. These studies are categorised and reviewed based on the specific techniques as follows.

GP with feature selection: In [244], an enhanced cooperative coevolution GP with feature selection was proposed. This method initially runs cooperative coevolution GP to obtain a diverse set of promising scheduling heuristics. Subsequently, feature selection and construction are performed based on the obtained heuristics to generate high-level terminals, which are then added to the terminal set. The cooperative coevolution GP is then rerun using the extended terminal set to evolve high-quality scheduling heuristics. In [269], an improved cooperative coevolution GP with feature selection was developed. Feature selection is employed to remove unimportant features and reduce the search space. Similarly, the same feature selection technique is utilised in multi-tree GP in [246]. In [255], a novel

two-stage multi-tree GP with feature selection was designed to evolve scheduling heuristics specifically with the selected features for DFJSS. Meanwhile, individual adaptation strategies were proposed to leverage the information from both the selected features and the investigated individuals during the feature selection process. In [256], a feature selection strategy was developed to guide GP in searching for the more promising region by utilising information during the evolutionary process. Specifically, the probability of each feature is calculated based on the number of occurrences of features. A larger number of occurrences result in a higher probability for the feature. During the generation of new individuals and subtrees for mutation, features are selected based on their probability. In [257], novel feature selection strategies were proposed to guide subtree selection for crossover and mutation rather than selecting them randomly. The occurrences of features are utilised to measure the importance of each subtree of the selected parents. The probability of selecting a subtree is based on its importance and the type of genetic operators. In [258], a correlation coefficient-based recombination guidance for GP was proposed. The importance of subtrees is measured based on the correlation coefficient, and the probability of subtree selection is set according to the importance of the subtree. During crossover, offspring is generated based on the probability of a subtree, meaning an unimportant subtree from one parent will be replaced by an important subtree from the other parent. These studies focus on integrating feature selection techniques into GP to guide its search towards promising regions, thereby evolving high-quality scheduling heuristics while saving training time.

GP with surrogate: In [266], appropriate surrogates for cooperative co-evolution GP were investigated to reduce its computation time while not sacrificing its performance for solving the DFJSS problem. With the experimental results, the potential of using surrogate models for solving the DFJSS problem was confirmed. The effectiveness of surrogate models in terms of accelerating the evolutionary process and improving the qual-

ity of the evolved scheduling heuristics on both cooperative coevolution GP and multi-tree GP was further verified in [283]. In [259], an effective collaboration mechanism with knowledge transfer for multi-fidelity-based surrogate models for GP was proposed to solve the DFJSS problem. First, depending on the characteristics of the problem, the given problem was simplified and several surrogate models with different fidelity were developed. Second, an effective collaborative framework with knowledge transfer was proposed for the designed multi-fidelity-based surrogate model to evolve effective scheduling heuristics. Regarding knowledge transfer, this study uses the subtree transfer strategy, wherein a parent is chosen from other subpopulations during crossover based on the transfer rate.

GP with multi-task learning: In [254], a task relatedness-based multi-task GP was proposed. This study first develops a new relatedness measure between tasks for multi-tree GP, then proposes a new multitask GP algorithm that adaptively selects assisted tasks based on the relatedness measure to solve the DFJSS with multiple tasks simultaneously. The surrogate models are also used in GP for solving multi-task DFJSS problems [263, 264]. For each task, a surrogate is built which is used not only to improve the efficiency of solving every single task but also for knowledge transfer between different tasks.

Moreover, various GP variants find extensive application in addressing the DFJSS problem with multiple objectives [39, 260, 271]. Several investigations [267, 282, 284] integrate cooperative coevolution GP and multi-tree GP with NSGAII and SPEA2 to evolve a Pareto front of scheduling heuristics for solving MO-DFJSS problems, often involving new jobs' dynamic arrival events. In [207], the MO-DFJSS problems incorporating both new job arrivals and machine breakdowns are explored. These approaches typically make routing and sequencing decisions in a delay-free manner, which might have limitations in dynamic environments. To address this, [236] proposed a delay method for the multi-objective DFJSS

problem, delaying routing decisions to ensure decisions are made under the latest and most accurate information. Specifically, ready jobs are sent to a pool instead of immediately assigning them to machines. When a machine becomes idle, the delayed routing policy is employed to create a candidate operation set for that idle machine. In [271], an interpretability-aware multi-objective GP was proposed, optimising not only for scheduling quality but also for smaller heuristic sizes to enhance interpretability. Some studies simultaneously handle multiple tasks and multiple objectives in DFJSS [261, 270].

Overall, the studies discussed above demonstrate the effectiveness of GP in addressing DFJSS problems. Subsequently, the following section reviews related works on GP and specific techniques employed in this thesis.

2.5 Related Work

Based on the above research, scheduling heuristics have been widely used to solve JSS problems, which can produce good enough solutions within an acceptable time limit. Also, the scheduling heuristics can be easily applied to practical problems. At the same time, GP, as a hyper-heuristic, can automatically evolve high-quality scheduling heuristics. Therefore, this section provides an in-depth investigation of studies on GP for evolving scheduling heuristics, which is also the focus of this thesis. In addition, related works about the specific techniques used in this thesis contributing to the enhancement of GP are also reviewed.

2.5.1 Parent Selection in GP

In standard GP, tournament selection [26] stands out as the most commonly employed parent selection method. This technique involves randomly sampling k individuals (where k is the tournament size parameter) from the population. Subsequently, the individual with the best fitness

among the sampled group is chosen as the parent. Adjusting the k parameter allows for control over the selection pressure, where a larger k value results in higher pressure. Notably, when $k = 1$, tournament selection simplifies to random selection. Many studies have explored tournament selection in depth, examining its behaviour and seeking ways to improve its effectiveness. Early studies focus on sampling strategies and selection pressure.

Fine-tuning the tournament size represents the most direct strategy for controlling selection pressure in tournament selection [148]. In a mathematical analysis of tournament selection conducted in [27], insights into the behaviour of tournament selection and the impact of the tournament size are provided. However, adjusting the tournament size only enables coarse-level control over selection pressure. An alternative strategy was proposed in [80], introducing an additional probability to control the selection pressure at a fine level. In [148], the selection pressure of tournament selection under noisy environments was studied and the strong effect of selection pressure on population diversity and convergence rate is emphasised. Additionally, [200] proposed an unbiased tournament selection to mitigate sampling bias and eliminate diversity loss. A comprehensive exploration of selection pressure control in GP is investigated in [230]. This study proposed an automatic selection pressure control strategy for GP to dynamically adjust the individuals considered as candidates for tournament selection. Results indicate that this selection pressure control strategy could enhance both the effectiveness and efficiency of GP. Xie et al. [231] investigated the impact of selection pressure on performance by comparing standard parent selection with and without selection pressure, revealing a significant improvement in algorithm performance with a level of selection pressure. In [232], Xie et al. studied the impact of the tournament selection with no replacement, and the results show that tournament selection with no replacement does not make the selection behaviour significantly different from that in the standard one. Further-

more, Xie et al. [229] developed a novel strategy integrating knowledge of the fitness rank distribution into tournament selection to dynamically adjust selection pressure. The results indicate the effectiveness of this new strategy, showcasing the potential of leveraging fitness rank distribution knowledge for dynamic selection pressure adjustment.

Various extensions of tournament selection have been proposed, introducing alternative selection schemes. Cooperative selection [109] and rank-based selection [201] represent two such variations, selecting parents not solely based on fitness but incorporating some rank principles. Semantics have been integrated into tournament selection [70] to enhance semantic diversity. Semantic-based tournament selection identifies individuals with strong performance but different semantics (behaviours), yielding promising results in certain regression problems. In [47], a novel form of tournament selection was developed using statistical analysis of GP error vectors. Results demonstrate that these new selection schemes contribute to improved semantic diversity, outperforming both standard tournament selection and semantic-based selection [70] in regression problems.

Some studies focus on parent selection for crossover. Comparative partner selection [50] aims to choose a pair of complementary parents, emphasising strengths on different training instances. However, this method might overlook high-quality parent pairs due to its underdeveloped selection strategy. Diverse partner selection [7] addresses the limitations of comparative partner selection by proposing a smarter strategy. The above methods tend to select *generalist* parents that perform relatively well on all the training instances and exhibit different behaviours, showcasing their effectiveness in generating high-quality offspring. An alternative strategy involves selecting “specialist” individuals that excel on specific training instances despite having lower overall fitness. This allows for the utilisation of special promising building blocks (e.g., sub-trees) from these individuals, which might be lost in standard parent selection schemes. To retain the promising information in the specialist individuals, Helmuth

and Matheson [92] proposed the lexicase selection. The basic principle of lexicase selection is to select parents based on each training case separately rather than an aggregated function such as mean squared error. For example, given an order of the training cases, lexicase selection selects the individuals in the population that perform the best on the first case and moves on to the next case. This step is continued until only a single individual is selected. This way, a specialist individual has a chance to be selected if its specialised cases are ordered first.

Lexicase selection [203] has outperformed tournament selection across some benchmark problems [90, 123, 208]. The principle of lexicase selection is to remove the poor individuals from the whole population one by one based on a random sequence of the fitness cases/training instances, until only one individual remains (ties are broken randomly), which is the selected parent. For each fitness case, the individuals performing worse than the best individual in this case will be removed. Based on this, many different variants of lexicase selection methods are proposed, including ϵ -lexicase selection, random threshold lexicase selection, MAD-CAP ϵ -lexicase selection, and truncated lexicase selection [203]. They use different strategies to remove poor individuals.

Among them, ϵ -lexicase selection [123] stands out as the most effective lexicase selection strategy for parent selection. In ϵ -lexicase selection, the criteria for eliminating individuals are somewhat relaxed compared to traditional lexicase selection methods. Specifically, an individual can still be retained if its fitness, denoted as $fit(x)$, is not significantly worse than the fitness of the best individual, denoted as fit^* , in the current fitness case, i.e., $fit(x) < fit^* + \epsilon$. The parameter ϵ is determined based on the entire population and might vary across generations. Overall, ϵ -lexicase selection is the method that can obtain the best solutions among the different variants of lexicase selection methods and is often used as a basis to propose new variations. In [180], ϵ -lexicase selection is used to solve the DFJSS problem for the first time, in which the training set is designed with

many different instances by using different numbers of jobs and machines. However, they only consider small-scale instances, where the maximum number of jobs is no more than 100, and they obtain similar results with tournament selection. For evaluating the *steady-state* performance, a large-scale simulation with thousands of job arrivals is often required. In this case, the fitness evaluation for an instance becomes very time-consuming, making it not affordable to use a large number of training instances as fitness cases. Hence, the application of ϵ -lexicase selection to solve the DFJSS problem without an increase in computational time requires the development of new strategies.

2.5.2 Ensemble GP

Ensemble learning is a widely used technique in the machine learning community [211, 272]. Ensemble learning [189] involves combining the decisions of multiple models to improve the overall performance of the system. The idea is to leverage the diversity and complementarity of different models to create a stronger and more accurate ensemble model than each element model. Ensemble GP is a technique that combines the power of GP with the idea of ensemble learning [187]. Recently, ensemble GP has gained attention in the research community for learning a group of scheduling heuristics [41].

In [88], NELLI-GP is proposed by extending a classical ensemble method called NELLI, to evolve an ensemble of scheduling heuristics for solving the JSS problem. NELLI-GP adopts a divide-and-conquer strategy in which each scheduling heuristic in the ensemble solves a unique subset of instances. Experiments show that an ensemble of scheduling heuristics can be evolved that outperforms existing scheduling heuristics and recent hyper-heuristic methods. In [176], an ensemble of scheduling heuristics is evolved using cooperative coevolution GP for solving the static JSS problem and this method can produce more robust scheduling heuristics than

the classical GP. The cooperative coevolution method contains the same number of subpopulations as the number of elements in the ensemble, and each subpopulation is used to evolve a single scheduling heuristic for the ensemble. However, these works only consider a static scheduling environment. Further, the individuals (scheduling heuristics) from different subpopulations interact with each other only when they are evaluated together as an ensemble.

Numerous studies have explored the application of ensemble methods in dynamic scheduling problems. In [211], four ensemble learning methods are investigated for creating ensembles of scheduling heuristics to solve dynamic scheduling problems. These methods included simple ensemble combinations, BagGP, BoostGP, and cooperative coevolution GP. The results indicate that creating ensembles of scheduling heuristics through simple ensemble combinations, BagGP, and BoostGP led to improved scheduling results compared to the standard GP method. In [177], a preliminary investigation into using GP to evolve ensembles of scheduling heuristics for solving the dynamic JSS problem is conducted. The grouping is done randomly, and multiple groupings are made for a single individual to assess overall performance within an ensemble. However, the proposed method does not show significantly better performance than the standard GP method. Certain studies have investigated the comparison of different aggregation methods for forming ensembles [175, 212]. In [212], the comparison of various aggregation methods reveals that the voting aggregation method is notably more stable than other aggregation methods. In [172], the authors applied ensemble GP and multilevel GP to solve the dynamic JSS problem, demonstrating that using ensembles could improve performance compared to single scheduling heuristics.

In addition to these studies focusing on classical ensemble learning methods, some propose innovative ideas. In [209], the authors explore various ensemble learning methods and propose an ensemble subset selection method to eliminate elements from the ensemble that do not con-

tribute to its quality, utilising extra training instances. However, the use of extra training instances requires additional data and makes the evaluation more time-consuming. In [171], the authors apply the niching mechanism to cooperative coevolution GP to evolve an ensemble for dynamic JSS. Fitness is determined by performance and diversity within an ensemble, calculated through phenotypic distance. The results show that the niched cooperative coevolution GP method achieved comparable performance to baseline cooperative coevolution GP methods but with smaller rule sizes. In [77], a novel ensemble construction method is proposed, allowing ensembles to include not only scheduling heuristics but also other ensembles. While this method can yield better performance, it comes at a higher computational cost.

Some studies explore the combination of GP with heuristic algorithms, initially learning a set of scheduling heuristics through GP and subsequently learning ensembles using heuristic algorithms [74, 75, 78]. For instance, in [75], a hybrid algorithm that integrates GP and the genetic algorithm is proposed to evolve an ensemble of scheduling heuristics for solving an online one-machine scheduling problem. This algorithm is a two-stage algorithm, with GP generating scheduling heuristics and the genetic algorithm evolving ensembles from the heuristics produced by GP. While this method allows the exploration of different ensemble combinations through the genetic algorithm, it comes with additional training time. Moreover, it focuses solely on the one-machine scheduling problem, which is less complex than the DFJSS problem emphasised in this thesis.

In summary, the study in solving the scheduling problems with GP and ensemble learning is limited. Most of the studies are mainly about the investigation or comparison of classical ensemble methods applied to scheduling problems, which cannot provide significantly better performance but suggest that the voting aggregation method can support better stability than other aggregation methods. Moreover, the existing studies about combining GP with ensemble technique to scheduling

problems mainly focus on evolving scheduling heuristics independently, then grouping the evolved scheduling heuristics by some greedy selection strategies, while rarely considering the evolution with ensembles. To the best of our knowledge, no study about GP considers evolution with single individuals and ensembles together in the research area of DFJSS. By allowing breeding between single individuals and ensembles, the search space can be further explored, increasing the likelihood of discovering novel and more effective scheduling heuristics. To fill this research gap and improve the performance of GP for DFJSS, an effective ensemble GP method is required.

2.5.3 RL for JSS

RL-based approaches have recently attracted much attention for addressing diverse scheduling problems, either through indirect or direct approaches [98]. In an indirect approach, RL agents are combined with manually designed scheduling heuristics. Conversely, the direct approach involves extracting state features by observing the environment and generating a scheduling scheme directly with the agent, often referred to as “end-to-end”. End-to-end RL approaches are primarily employed in static scheduling problems or scenarios where the number of jobs/machines is predetermined. The action space remains constant, set to the same number as the jobs/machines, across different decision points. For instance, in [278], the actions directly represent candidate machines, while in [250], the actions correspond to candidate jobs. In [125], an innovative end-to-end deep RL is designed to tackle the scheduling problem. It demonstrates promising results, but it still encounters challenges when applied to more complex dynamic scenarios. Following [125], the same authors further propose a new end-to-end deep RL method for DFJSS [124]. The main principle is to divide the DFJSS problem into multiple static scheduling problems. However, the inherent essence of such an approach is still to

view the dynamic problem as a static problem to be solved. This does not allow for timely response to changes in the scheduling environment and real-time decision-making, which ultimately affects the quality of the solution. Moreover, such an approach is not efficient. The advantage of end-to-end methods lies in minimising the need for human-designed heuristics. This approach allows the model to learn decision-making directly from raw observations. However, a limitation of these methods is their inability to handle problems with dynamically changing action spaces, such as DFJSS. In DFJSS, the number of jobs can vary in response to a dynamic environment. Consequently, the number of available actions of deep RL cannot remain constant throughout, as it needs to adapt to the requirements of changing jobs in the scheduling system.

In the indirect application of RL to scheduling problems, manually designed scheduling heuristics are commonly employed as the agents' actions. These RL methods address the challenges faced by end-to-end RL approaches, which encounter difficulties with dynamically changing action spaces. Instead of directly utilising machines/operations as actions, these methods employ manually designed rules as actions. In [249], a deep RL method with a DQN is proposed for the dynamic JSS problem. This paper specifically focuses on determining the sequencing rule, and the actions employed consist of widely used manually designed sequencing rules. In [56], a deep RL method with a DQN is proposed to solve a multi-objective flexible JSS problem with crane transportation and setup times. To tackle the two subproblems within flexible JSS, i.e., operation sequencing and machine routing, this paper integrates manually designed composite scheduling heuristics, combining these subproblems into a unified framework. Then an agent is learned to make decisions among these composite scheduling heuristics. Several other studies also adopt a similar approach by manually designing composite scheduling heuristics as actions for RL to solve the DFJSS problem [36, 139, 227, 275]. However, it is crucial to acknowledge that, for these studies, the strategy of combining

the two subproblems is typically not optimal. It cannot fully capture the interaction between the routing and sequencing rules [265], which could substantially reduce the search space, potentially constraining the exploration of superior solutions.

In addition to the aforementioned methods that directly utilise manually designed composite scheduling heuristics as actions, some approaches leverage RL to learn the weights associated with these manually designed composite scheduling heuristics, resulting in a weighted scheduling heuristic [84]. A similar but more advanced strategy is proposed in [45], where GP is used to learn scheduling heuristics, replacing the manually designed ones. This kind of method can be effective in adapting to dynamic and changeable environments, as it does not necessitate consideration of the impact of variations in the number of candidate machines/operations at different decision points. However, a notable limitation of this method is the pre-determination of the aggregation function, which solely relies on a weighted sum function. This predetermined choice still narrows down the search space by limiting the range of functions available. Additionally, no sufficient evidence is provided to explain why this particular predetermined aggregation function is able to yield good results.

In [136], a deep RL method with a double DQN model is proposed to solve the DFJSS problems. Different from the aforementioned studies utilising composite scheduling heuristics as actions to jointly address sequencing and routing decisions, they employ a distinct training strategy. Specifically, the sequencing rule and the routing rule are trained separately. For training the routing rule, the actions directly correspond to the machines, while for the sequencing rule, manual rules are used as actions, which can address the challenge posed by the varying number of jobs in different decision-making scenarios. This method would not reduce the search space as much as in the above studies. In such cases, deep RL is utilised to enhance the performance of traditional methods but

cannot fully address their inherent limitations. For instance, if the manually designed scheduling heuristics lack high quality, their effectiveness as actions can be constrained, and the incorporation of RL might not yield substantial improvements.

Following the work [136], to overcome its limitation that needs a lot of domain knowledge of experts to design scheduling heuristics, a possible way is to use the heuristic generation method (e.g., GP) to automatically learn some high-quality and diverse scheduling heuristics as actions. Therefore, it is reasonable and necessary to explore the combination of GP and RL in solving the DFJSS problem. The key advantage of using GP lies in its ability to automatically explore a wide range of potential scheduling heuristics and adapt to the specific characteristics of the problem at hand with little domain knowledge [206]. Traditional GP methods typically focus on obtaining the best scheduling heuristic without emphasising the diversity within the population. However, in addition to the effectiveness of each scheduling heuristic as an action, diversity among actions is crucial in RL. Niching [246] has proven to be an effective strategy employed in GP to enhance both its effectiveness and population diversity [143]. This is achieved by fostering the presence of multiple diverse scheduling heuristics within the population.

Drawing on the strengths of GP and the niching strategy, it would be interesting to investigate whether the performance of RL can be enhanced by employing GP with a niching strategy to autonomously learn effective and diverse scheduling heuristics as actions to address the DFJSS problems. Furthermore, it would also be interesting to investigate whether this integration can enhance the performance of GP for DFJSS.

2.5.4 Multi-objective GP

DFJSS, aiming to address multiple objectives simultaneously, poses a significant challenge. Presently, there are studies exploring the application

of Pareto dominance-based algorithms for MO-DFJSS problems [267, 282]. In [267], strategies from non-dominated sorting genetic algorithm II (NSGAII) [52] and strength Pareto evolutionary algorithm 2 (SPEA2) [289] are integrated for the first time into GP, named NSGP2 [267] and SPGP2 [267], with a multi-tree representation to evolve scheduling heuristics for solving MO-DFJSS. In [282], not only is tree-based GP with NSGAII and SPEA2 tested, but cooperative coevolution GP with NSGAII and SPEA2 is also proposed to address the MO-DFJSS. These studies have demonstrated that NSGAII and SPEA2 can be combined with GP to produce high-quality Pareto fronts of scheduling heuristics in terms of HV and IGD for tackling MO-DFJSS. However, besides HV and IGD, the spreadability of the Pareto front and the consistency in the behaviours of the scheduling heuristics are also important and need to be considered.

The MOEA/D is a well-known multi-objective method [276]. MOEA/D decomposes a multi-objective Problem into several sub-problems using a scalar function and then optimises them simultaneously [85]. MOEA/D possesses strengths that make it potentially capable of evolving a Pareto front of scheduling heuristics with good spreadability. MOEA/D utilises a decomposition approach that enables the optimisation of multiple sub-problems in different directions simultaneously, facilitating the generation of a spread Pareto front. Further, the high search capability of MOEA/D, especially for difficult multi-objective problems, has been repeatedly reported [104, 105, 126].

While there have been studies demonstrating the potential of using decomposition-based methods for solving static JSS problems [37, 108, 167, 235, 279], to the best of our knowledge, there is no study using MOEA/D for solving the MO-DFJSS problem. In this context, it is intriguing to investigate whether incorporating MOEA/D with GP can evolve a Pareto front of superior scheduling heuristics compared to the current state-of-the-art multi-objective algorithm for MO-DFJSS.

Moreover, beyond the innovation of integrating classical multi-

objective algorithms with GP and handling the challenges inherent in this integration, there is a need for additional exploration of other techniques. NSGP-II stands as the current state-of-the-art multi-objective optimisation algorithm for MO-DFJSS. Several studies further enhance NSGP-II in solving MO-DFJSS problems. In [260], a novel NSGP-II approach is presented for MO-DFJSS by incorporating surrogate techniques and brood recombination techniques. Through the utilisation of the surrogate and brood recombination-assisted method, the enhanced NSGP-II achieves superior scheduling heuristics compared to the original NSGP-II within the same training time. In [270], the impact of terminal settings on NSGP-II for solving MO-DFJSS is investigated. While some studies focus on interpretability [271] or multitasking [261] aspects in MO-DFJSS, the integration of other techniques in GP remains limited. Exploring additional techniques is essential to augment the capabilities of existing multi-objective GP algorithms in addressing the MO-DFJSS problem.

Given that existing multi-objective GP algorithms focus solely on the genotype of individuals, neglecting phenotype information that reflects genotype behaviour, a promising direction for improvement involves enhancing population diversity by integrating semantic techniques into these algorithms. Semantic techniques involve associating meaning with individuals. Semantic GP [217] has recently gained significant attention in the field of GP. It represents a valuable approach for incorporating semantic information into the evolution process, thereby improving the performance of evolved solutions. One of the key advantages of semantic GP is its ability to consider the behaviours/semantics rather than the genotype of individuals [217]. By considering the behaviour of individuals, semantic GP enables a more nuanced understanding of the evolved solutions. The semantic analysis facilitates the discovery of individual relationships and population composition, making semantic GP particularly valuable in domains where different genotypes can give the same behaviour, such as MO-DFJSS.

Most semantic GP methods are based on the usage of genetic operators that act on the genotype to produce offspring, and then accept offspring that satisfy some semantic criteria into the next population [217]. The semantic criteria can be semantic diversity [19, 20, 21, 42, 43, 69] and semantic similarity [44, 214, 215]. The consideration of semantic information enhances the exploration of different dimensions of search space. Semantic GP has demonstrated its effectiveness across various problem domains, including symbolic regression [214], classification [15, 192], and feature selection [169]. It would be interesting to explore whether the integration of semantic information into existing multi-objective GP algorithms can contribute to the evolution of effective Pareto fronts of scheduling heuristics for solving MO-DFJSS problems.

2.6 Chapter Summary

This chapter begins by introducing various types of JSS problems, with a specific emphasis on the DFJSS problem under investigation. Subsequently, it covers the foundational concepts of machine learning and evolutionary computation, which serve as essential background knowledge for this thesis. Following this, the technical aspects of GP, the distinction between heuristics and hyper-heuristics, the application of scheduling heuristics for solving DFJSS, and the technical intricacies of RL are discussed. Furthermore, the chapter explores various approaches to JSS, including exact methods, heuristic approaches, and hyper-heuristic approaches. Additionally, this chapter presents a comprehensive overview of related work in the field, beginning with studies on using GP for evolving scheduling heuristics, with a focus on those that are highly relevant to the research objectives of this thesis. The limitations of the existing research are highlighted below.

- Firstly, in GP for DFJSS, the existing parent selection mechanisms primarily rely on fitness, lacking effectiveness in choosing parents

with diverse abilities or strong cooperation abilities for generating offspring.

- Secondly, while a collaborative ensemble of scheduling heuristics tends to outperform a single heuristic, the development of ensemble GP for evolving such groups for DFJSS is still in its early stages, which requires further exploration.
- Thirdly, in addition to GP, RL has gained popularity in DFJSS as a heuristic selection approach. Both GP and RL have their strengths and limitations. While GP excels in intelligently generating scheduling heuristics, relying solely on a single heuristic throughout the long-term scheduling process might not always be effective. On the other hand, RL is good for heuristic selection. However, manually designing scheduling heuristics as actions can be time-consuming and might not always lead to effective performance. Exploring the combination of intelligent heuristic generation and selection simultaneously by incorporating GP and RL presents an intriguing and unexplored topic worthy of investigation.
- Finally, recognising that some users might have multiple requirements, there is a need for increased attention to MO-DFJSS problems.

An in-depth analysis of existing work on DFJSS has revealed key gaps and limitations, motivating the development of new methods for solving the DFJSS problems effectively. Therefore, subsequent chapters will systematically address these challenges, covering enhancements in improving parent selection mechanisms, leveraging the collective strength of multiple scheduling heuristics for effective joint decision-making instead of relying on a single one, developing intelligent heuristics selection method for DFJSS, and enhancing GP's multi-objective problem-solving capabilities.

Chapter 3

Diversity-based Parent Selection Mechanisms

This chapter proposes three novel diversity-based parent selection mechanisms in GP to leverage individuals with diverse strengths as parents for generating high-quality offspring to evolve effective scheduling heuristics for DFJSS.

3.1 Introduction

Parent selection plays an important role in identifying promising individuals that carry good genes [90]. Parent selection recommends individuals as parents for crossover, mutation, and reproduction to generate offspring. Tournament selection [62] is the classical parent selection method that selects parents based on their fitness. Tournament selection selects the parents independently, without considering diversity and the relationship between the selected parents. As a result, it may select high-quality parents with very similar structures/behaviours, leading to premature convergence in GP [31, 43]. To overcome this limitation, a straightforward way is to select individuals with different behaviours as parents to generate diverse individuals. Thus, it is reasonable to propose a new parent

selection mechanism in GP to select individuals not only based on fitness but also with different behaviours as parents.

On the other hand, some selection methods, such as diverse partner selection (DPS) [7] and lexicase selection (LS) [147] have the advantage of selecting more diverse parents to improve the quality of produced offspring. Both DPS and LS are proposed to select parents that specialise in different instances. On one hand, an individual is likely to be selected to be a parent regardless of its overall fitness (e.g., average error over all the instances), as long as it performs well on some instances. This can increase the diversity of the selected parents and thus the generated offspring. On the other hand, since each parent must specialise on some instances, it is expected to have some promising building blocks that can be inherited by its offspring. In other words, DPS and LS can select a wider range of parents with different promising building blocks rather than randomly reducing the selection pressure. DPS and LS have shown success in solving some optimisation problems [7, 90, 92, 208]. Thus, it is reasonable to extend the GP with DPS and LS to evolve scheduling heuristics as well.

However, it is non-trivial to extend the GP with DPS or LS from its successful domains such as classification, program synthesis, and symbolic regression to evolving scheduling heuristics. In terms of DPS, its success relies on the characterisation of the behaviour of an individual in different cases. In existing studies, a case is naturally defined as a data point for classification or regression, and the behaviour in a case can be easily characterised as the classification/regression error on that data point (the smaller the better). However, they cannot be directly applied to evolving scheduling heuristics for DFJSS, which is not a supervised learning problem (i.e., there is no ground truth of each decision in DFJSS). In terms of LS, the evaluation of an instance in DFJSS is very time-consuming. In program synthesis and symbolic regression, an instance typically consists of a given input and its target output. Evaluating an instance requires applying the model only once to generate the predicted outputs given the inputs. How-

ever, in DFJSS, an instance is usually a large-scale DFJSS simulation with thousands of jobs. Evaluating such an instance requires applying the GP model many times (each at a decision situation, when a machine becomes idle or a job operation becomes ready to be processed) to generate a schedule given the dynamic job arrivals in the simulation. As a result, we cannot afford a large number of DFJSS instances (i.e., simulations). On the other hand, if we use too few instances, then the effectiveness of LS is negatively affected.

According to the above issues, this chapter proposes three novel parent selection mechanisms for selecting diverse individuals as parents to generate high-quality scheduling heuristics for DFJSS.

3.1.1 Chapter Goals

The goal of this chapter is to *develop GP with three novel parent selection mechanisms that apply cluster selection, DPS, and LS effectively in GP for generating high-quality and diverse scheduling heuristics without increasing complexity for DFJSS*. By developing and comparing these three parent selection mechanisms, we expect to investigate which way of parent selection can achieve superior scheduling performance. Specifically, this chapter has the following objectives:

1. Propose novel GP algorithms with three novel parent selection mechanisms (i.e., cluster selection, DPS, and LS) that consider not only fitness but also diversity or complementarity to generate high-quality offspring for solving the DFJSS problem effectively.
2. Develop a novel multi-case fitness evaluation strategy to address the time-consuming evaluation issue of each instance (simulation) in DPS and LS by a new definition of case-fitness, which creates multiple cases from a single simulation.
3. Analyse the effectiveness of the proposed algorithms in terms of the

quality of evolved scheduling heuristics and how the proposed algorithms influence the population diversity of GP.

3.1.2 Chapter Organisation

The rest of this chapter is organised as follows. Detailed description of the proposed algorithms is given in Section 3.2. The experimental design is provided in Section 3.3, followed by results and discussion in Section 3.4. Further analyses are conducted in Section 3.5. Finally, Section 3.6 concludes this chapter.

3.2 Proposed Algorithms

This section describes the proposed GP algorithms with the three novel parent selection mechanisms (i.e., cluster selection, DPS, LS). For each algorithm, the overall framework of the proposed algorithm is first introduced. Then, the key components and the detailed parent selection mechanisms are given.

3.2.1 GP with Cluster Selection

The Overall Framework

The overall framework of the proposed GP with cluster selection (GPCS) for DFJSS is shown in Figure 3.1. As with traditional GP algorithms, population initialisation, fitness evaluation, elitism selection, parent selection, reproduction, crossover, and mutation are all main processes. The proposed algorithm uses reproduction, subtree mutation, and tree-swapping crossover operators [265] to generate offspring. For reproduction, the parents are directly inherited to the next generation. For subtree mutation, a new subtree is randomly generated by sampling from the terminals and functions. We then randomly select a subtree from the parent and replace

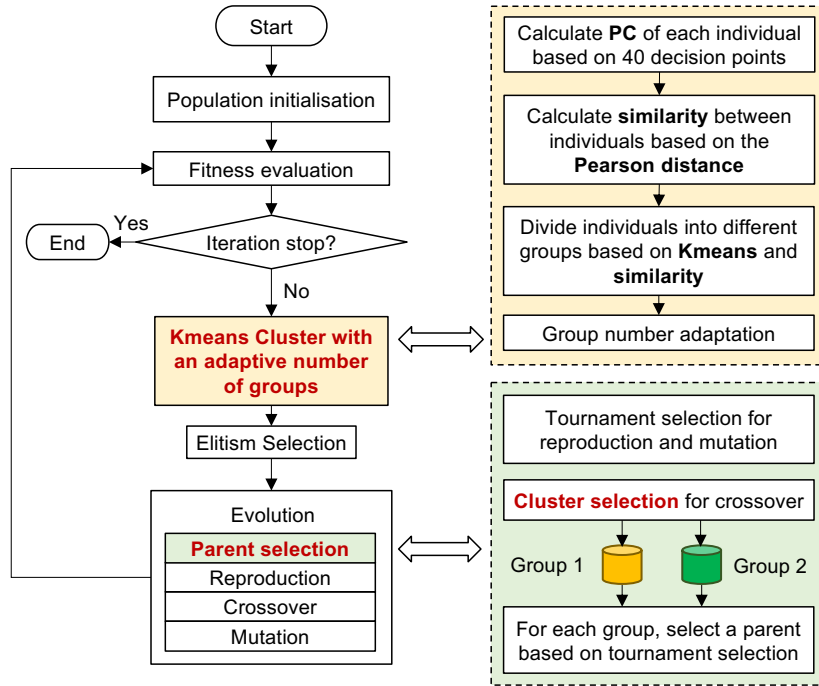


Figure 3.1: The flowchart of the proposed GPCS method.

it with the newly generated subtree. For tree swapping crossover, for one tree, we randomly select a subtree from each parent and swap them. For another tree, we simply swap the entire tree.

The main innovation of GPCS is to use cluster selection to replace tournament selection to select a pair of more diverse parents in the evolution process for crossover.

In the cluster selection, the Kmeans cluster strategy with an adaptive number of groups is proposed and used after fitness evaluation to divide individuals in the population into different groups based on their behaviour similarities. Each group contains a number of individuals with similar behaviours. Then, when two parents are needed for crossover, the cluster selection randomly selects two different groups, we call them *preliminary winners one* and *preliminary winners two*. After that, tournament selection is used to select two individuals from preliminary winners one

and preliminary winners two, respectively. These two individuals are the parents used for crossover.

Behaviour Similarity Estimation

For the cluster selection, the *decision priorities* on 20 sequencing decision points and 20 routing decision points [93] in a fixed instance is used to estimate how similar two individuals behave with each other. The selection of 20 decision points for both sequencing and routing balances accuracy and computational efficiency, informed by existing research [264, 263]. The *decision priority* denotes the index sequencing of the candidate operations/machines made by the individual (scheduling heuristic) on the decision point. For example, if we have 3 sequencing decision points and 3 routing decision points, each with 7 candidate operations/machines, and the index of candidate operation/machine is between 1 and 7. Then the decision priorities for an individual x on these decision points can be represented as Figure 3.2.


		Decision priority from high to low 						
Sequencing decisions	{	1	3	4	6	5	7	2
		2	3	1	5	4	6	7
		1	7	6	4	5	2	3
Routing decisions	{	6	2	5	7	4	3	1
		4	3	2	1	5	6	7
		4	1	3	5	7	6	2

Figure 3.2: An example of the *decision priorities* on 3 sequencing decision points and 3 routing decision points.

It can be seen from Figure 3.2, that each row represents the decision

priority on each decision point which is the priority order of candidate operations/machines. The whole table denotes the decision priorities of the individual x . We can estimate the behaviour similarity between two individuals based on the mean *Pearson distances* between decision priorities of the two individuals x, y . The *Pearson distance* on the decision point d_i is defined as the $dis(x, y)_i = 1 - \rho(x, y)_i$, where $\rho(x, y)_i$ represents the *Pearson correlation* which can be calculated by Eq. (3.1).

$$\rho(x, y)_i = \frac{\sum_{j=1}^n (X_{ij} - \mu_{X_i})(Y_{ij} - \mu_{Y_i})}{\sqrt{\sum_{j=1}^n (X_{ij} - \mu_{X_i})^2} \sqrt{\sum_{j=1}^n (Y_{ij} - \mu_{Y_i})^2}} \quad (3.1)$$

where n is the number of candidate machines/operations. $X_i = [X_{i1}, \dots, X_{in}]$ and $Y_i = [Y_{i1}, \dots, Y_{in}]$ represent the decision priority on the decision point d_i (each row in Figure 3.2) by two individuals, respectively. μ_{X_i} and μ_{Y_i} are the average value of X_i and Y_i .

Overall, the process of calculating the behaviour similarity between individuals is as Algorithm 1. Firstly, for each individual and each decision point, this method calculates the priority of each candidate, to form the *decision priority* (Line 2). Secondly, for each decision point, this method calculates the *Pearson correlation* between the *decision priority* of the two individuals (Line 3). At the same time, we can calculate the *Pearson distance* on each decision point (Line 4). Finally, the behaviour similarity is calculated as the mean of the *Pearson distance* over all the decision points (Line 6).

The Pearson correlation has been used in DFJSS to measure the importance of each subtree for an individual and has shown success in improving the effectiveness of the scheduling heuristics by swapping the unimportant subtree with the important subtree to do crossover [258]. Therefore, we use the mean Pearson distances to estimate the similarity between individuals. Based on Eq. (3.1), the Pearson correlation $\rho(x, y)$ is always in the range of $[-1, 1]$, so the mean Pearson distance is always between 0 and 2. To be specific, if two individuals have the same decision priority

on all the decision points, the mean Pearson distances between them is 0, on the other hand, if they have very different decision priorities on all the decision points, the mean Pearson distance between them is near 2.

Algorithm 1: Behaviour similarity estimation

Input: The two individuals: x and y ; 40 DFJSS decision points:

$$D = \{d_1, \dots, d_{40}\}.$$

Output: The behaviour similarity between the two individuals: $\text{sim}(x, y)$.

```

1 for  $i = 1 \rightarrow 40$  do
2   Apply the two individual  $x, y$  on the decision point  $d_i$  to get the decision
   priority  $X_i$  and  $Y_i$ , respectively;
3   Calculate the Pearson correlation  $\rho(x, y)_i$  between the decision priorities  $X_i$  and
    $Y_i$  of two individuals;
4   Calculate the Pearson distance  $\text{dis}(x, y)_i = 1 - \rho(x, y)_i$  between the two
   individuals;
5 end
6  $\text{sim}(x, y) = \sum_{i=1}^{40} \text{dis}(x, y)_i / 40$ ;
7 return  $\text{sim}(x, y)$ ;

```

Kmeans Cluster

The proposed cluster selection uses the Kmeans method to divide the population into different groups. Different from the traditional Kmeans method which uses the Euclidean distance to estimate the similarity between two lists of values, we use the mean Pearson distance to estimate how similar two individuals behave with each other, which is introduced before. The center for each group is selected based on fitness. That is, the decision priorities of the individual with the best fitness in a group is set as the center.

In the Kmeans method, pre-setting the group size may result in uneven distribution, with some groups containing a substantially larger number of individuals compared to others. For instance, one group may have 100 individuals, while another might have only 5. This imbalance poses a chal-

lenge, as selecting a parent from a smaller group may lead to the hyper-selection (selected many times) of certain individuals with good fitness, potentially compromising the diversity of selected parents. To address this issue, we introduce an adaptive cluster number strategy. After the Kmeans method partitions the population into distinct groups, we examine the number of individuals in each group. If a group contains fewer than m individuals, those individuals are relocated to another group with the most similar behaviour. The similarity is calculated using the mean Pearson distance between each individual and the central individual in the target group. Subsequently, groups devoid of individuals are eliminated. This adaptive approach aims to maintain diversity in parent selection by mitigating the impact of unevenly sized groups.

The Cluster Selection

The cluster selection is used to select parents only for crossover. Different from the traditional tournament selection, it has two steps, the first step is to select two groups randomly. Then, in each group, the tournament selection is applied to select a parent with the best fitness in the second step. In this way, we can select two parents with promising fitness and perform dissimilar to each other.

3.2.2 GP with Diverse Partner Selection

The Overall Framework

The overall framework of the proposed GP with DPS (GPDPS) for DFJSS is shown in Figure 3.3. At first, a population of individuals is initialised by the ramped-half-and-half method. Then, at each generation, each individual is evaluated by the newly developed multi-case fitness evaluation. The multi-case fitness evaluation calculates both the standard fitness value for other genetic operators (e.g., mutation and elitism) and a list of

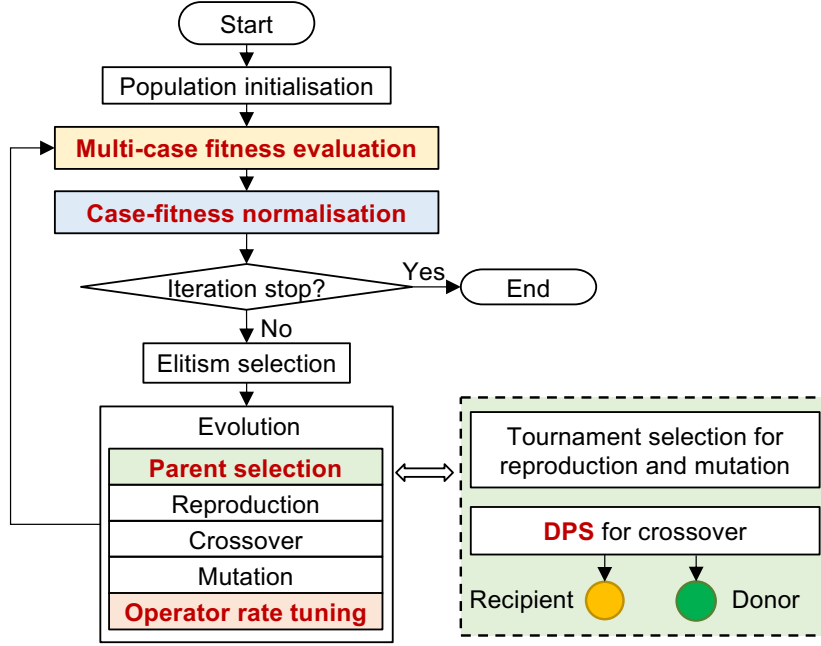


Figure 3.3: The flowchart of the GPDPS method for DFJSS.

case-fitnesses that play the role of the bit string in DPS for crossover. After the multi-case fitness evaluation, the case-fitness values in the list are normalised by the fitness normalisation step. Then, a number of elites with top-standard fitness are selected for the next generation directly. Afterward, the breeding process is conducted to generate offspring through parent selection and genetic operators. Specifically, the parents for mutation and reproduction are selected by the traditional tournament selection, while the parents for crossover are selected by the newly developed DPS operator based on the normalised list of case-fitness calculated by the multi-case fitness evaluation. Note that the crossover and mutation rates are adjusted based on the number of times that the suitable second parent fails to be selected by DPS. The operator rate tuning strategy is the same as the original method [7]. Specifically, each time DPS fails to select suitable parents, the crossover (mutation) rate is decreased (increase) by $1/popsiz$. Because if DPS consistently fails to select suitable parents, it suggests the

current crossover and mutation rates might not be promoting sufficient exploration and diversity within the population. By decreasing the crossover rate and increasing the mutation rate, the algorithm prioritises exploration over exploitation. This allows for the introduction of new genetic material and potentially helps the population increase diversity. After the entire breeding process, the crossover and mutation rates are reset to the default values. The above steps are repeated until a stopping criterion is met, and the best individual is returned.

In the proposed GPDPS for DFJSS, we adopt the multi-tree individual representation and multi-tree-based genetic operators, which are the same as that of GPCS. Briefly speaking, each individual consists of two trees, one representing the routing rule, and the other representing the sequencing rule. Given two parents, the crossover randomly selects one tree, and swaps random subtrees between this tree of the two parents. It also swaps the entire other tree of the two parents. Given a parent for mutation, we randomly select a tree from the parent and replace a random subtree of the tree with a newly generated subtree. More details can be found in the original literature [265].

Overall, the main differences between GPDPS and existing GP methods are the steps of (1) multi-case fitness evaluation, (2) case-fitness normalisation, (3) parent selection for crossover based on DPS, and (4) operator rate tuning, which are highlighted in Figure 3.3. They will be described in more detail subsequently [265].

Multi-case Fitness Evaluation

In DFJSS, one is typically interested in the “steady-state” performance of a scheduling heuristic, i.e., when the job shop state such as utilisation level becomes stable in the simulation. To this end, the DFJSS simulation needs to have a substantially large number (e.g., 5000) of job arrivals to be able to exclude the warm-up stage (e.g., the first 1000 job completions) and measure the “steady-state” performance of scheduling heuristics. As a result,

even a single simulation is time-consuming. To balance training efficiency and generalisation, most existing GP approaches [94, 236, 264, 283] use a single simulation for fitness evaluation in each generation, but change to a new simulation (e.g., by changing the random seed) in each new generation. Specifically, the evaluated scheduling heuristic is applied to the simulation/instance to generate a schedule, and its fitness is set to the objective value (e.g., mean flowtime, maximal tardiness) of its generated schedule. If considering each simulation (the case-fitness is the objective value of the generated schedule), then the traditional fitness evaluation with a single simulation is not applicable for DPS, since there are not enough number of instances. To address this issue, we propose a new multi-case fitness evaluation strategy that can extract a large number of cases from a single simulation.

Algorithm 2: Multi-case fitness evaluation for DFJSS.

Input: The individual to be evaluated: x ; DFJSS simulation: sim ; Number of jobs in the simulation: n ; Number of cases: c .

Output: Standard fitness $sf(x)$; Case-fitness vector $\mathbf{cf}(x)$.

- 1 Calculate the number of jobs in each case $g = n/c$;
 - 2 Run the simulation sim with the scheduling heuristic x to obtain the corresponding schedule $S = sim(x)$;
 - 3 **for** $j = 1 \rightarrow n$ **do**
 - 4 | Obtain the job completion time C_j from S ;
 - 5 **end**
 - 6 **for** $i = 1 \rightarrow c$ **do**
 - 7 | Set the i th group of jobs $\mathcal{J}_i \leftarrow \{g \times (i - 1) + 1, \dots, g \times i\}$;
 - 8 | Calculate the case-fitness $\mathbf{cf}_i(x) \leftarrow \text{obj}_{j \in \mathcal{J}_i}(C_j)$;
 - 9 **end**
 - 10 Calculate $sf(x)$ based on Eq. (3.4) or Eq. (3.5).;
 - 11 **return** $sf(x)$, $\mathbf{cf}(x)$;
-

The multi-case fitness evaluation strategy is described in Algorithm 2. Specifically, we divide the jobs to be processed in the simulation into a number of groups. First, we index the jobs in the order of their arrivals in

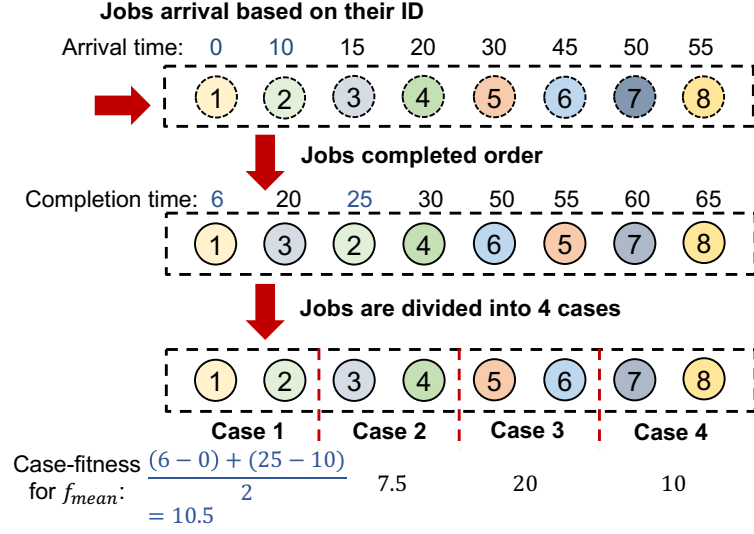


Figure 3.4: An example of dividing 8 jobs in a simulation into 4 groups.

the simulation, i.e., job 1 is the first arrived job. This way, different evaluated individuals have the same jobs in each case, and their case-fitnesses are comparable. Then, we create each case based on a subset of jobs. If there are total n jobs to be divided into c groups (assuming n can be divided by c), then each group contains $g = n/c$ jobs. Specifically, group i consists of the jobs with index from $g \times (i - 1) + 1$ to $g \times i$. Figure 3.4 gives an example of dividing 8 jobs in a simulation into 4 cases. Each job has an arrival time and a completion time. Note that the jobs might not be completed in the same order they arrive (e.g., job 3 arrives after job 2, but is completed before it). The 8 jobs are divided into 4 cases/groups based on their arrival order. Then, we can calculate the objective function based on the jobs in each case. For example, if the objective is the mean flowtime, then the case-fitness of case i is calculated by Eq. (3.2).

$$\text{cf}_i(x) = \frac{1}{g} \sum_{j=g \times (i-1)+1}^{g \times i} (C_j - r_j). \quad (3.2)$$

where C_j and r_j represent the completion time and arrival time of the job J_j .

If the objective is the maximal tardiness, then the case-fitness of case i is calculated by Eq. (3.3).

$$\mathbf{cf}_i(x) = \max_{j \in \{g \times (i-1) + 1, \dots, g \times i\}} T_j. \quad (3.3)$$

where T_j denotes the tardiness of job the J_j .

The standard fitness can be either calculated directly from the job completion time, or from the case-fitnesses. For max-objectives, the standard fitness can be calculated by Eq. (3.4). For mean-objectives, the standard fitness can be calculated by Eq. (3.5).

$$sf_{max} = \max \{\mathbf{cf}_1, \mathbf{cf}_2, \dots, \mathbf{cf}_c\}, \quad (3.4)$$

$$sf_{mean} = \frac{1}{c} \sum_{i=1}^c \mathbf{cf}_i. \quad (3.5)$$

Case-Fitness Normalisation

To avoid bias to any case during DPS, for each case, we normalise the case-fitnesses of all individuals in the population. In this context, bias refers to a situation where certain cases disproportionately influence the calculation of the benefit score. This means that some cases might have a greater impact on the final benefit score than others. Specifically, for each case $i = 1, \dots, c$, the normalisation is done by

$$\mathbf{cf}_i(x) \leftarrow \frac{\mathbf{cf}_i(x) - \min\{\mathbf{cf}_i(x') | x' \in pop\}}{\max\{\mathbf{cf}_i(x') | x' \in pop\} - \min\{\mathbf{cf}_i(x') | x' \in pop\}}. \quad (3.6)$$

The New Diverse Partner Selection

Algorithm 3 describes the newly developed DPS. It first selects the first parent, i.e., the *recipient*, by tournament selection. Then, it keeps selecting the second parent by tournament selection and examines if the second parent has sufficient positive influence on the recipient. To this end, we define α as the actual influence of the second parent to the recipient, and β as the

expected positive influence. If $\alpha > \beta$, then the second parent is confirmed as the *donor*, and the selection is terminated. Otherwise, the selection is continued. In the end, the operator rates are tuned accordingly.

Compared with the existing DPS, the main difference is on the calculation of the α and β parameters. Specifically, the new α parameter is defined as follows.

$$\alpha(x_1, x_2) = \sum_{i=1}^c \frac{\text{cf}_i(x_1) - \text{cf}_i(x_2)}{\max(\text{cf}_i(x_1), \text{cf}_i(x_2))}. \quad (3.7)$$

In other words, it calculates the total normalised advantage of the case-fitness $\text{cf}_i(x_2)$ over $\text{cf}_i(x_1)$ (the smaller the better, since the objective is minimised) over all the cases.

Algorithm 3: Diverse Partner Selection for DFJSS.

Input: The population: pop ; Number of trials K

Output: Selected parents: x_1, x_2

```

1 Select  $x_1 \in pop$  by tournament selection;
2 for  $k = 1 \rightarrow K$  do
3   | Select  $x_2 \in pop$  by tournament selection;
4   | Calculate  $\alpha(x_1, x_2)$  by Eq. 3.7;
5   | if  $\alpha(x_1, x_2) > 0$  then
6   |   | return  $x_1, x_2$ ;
7   | end
8 end
9 Decrease (increase) crossover (mutation) rate by  $1/popsize$ ;
10 return  $x_1, x_2$ ;

```

The new β parameter is set to 0, which means that the crossover is allowed if the second parent has a positive influence on the recipient.

Figure 3.5 gives an example of finding a suitable donor for the recipient. The left side shows the normalised case-fitness lists of three individuals, where *Ind1* represents the recipient and *Ind2* and *Ind3* are the candidates of the donor.

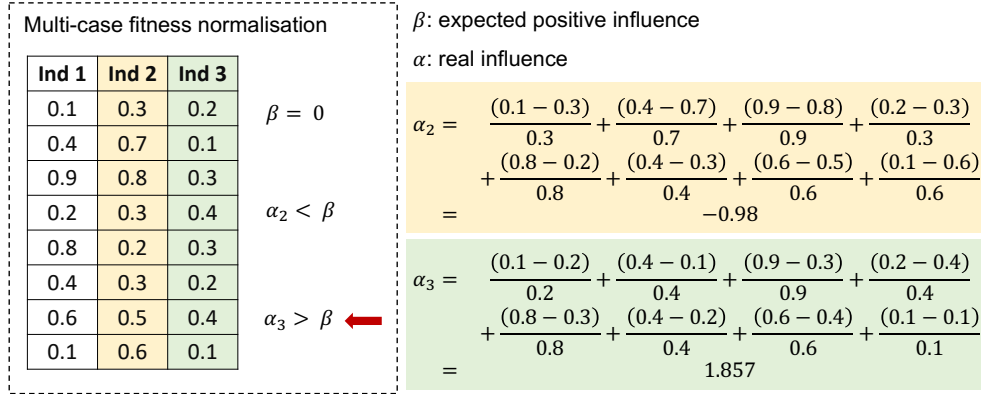


Figure 3.5: An example of the new DPS method.

The right-hand side calculates the α values of *Ind2* and *Ind3*. We can see that α_2 is negative while α_3 is positive. Therefore, we select *Ind3* rather than *Ind2* as the donor.

3.2.3 GP with Lexicase Selection

Overall Framework

Figure 3.6 shows the overall framework of GP with LS (GPLS). It starts with initialising a population of individuals, each as a scheduling heuristic. Then, in each generation, the newly proposed *multi-case fitness evaluation* is applied to each individual to obtain both the standard fitness (that standard GP uses) and a case-fitness vector. The multi-case fitness evaluation has been described in section 3.2.2. Afterwards, the elitism selection selects the elite individuals based on their standard fitness. Then, the *new LS* is applied to select parents to go through the breeding process and generate offspring by genetic operators. The breeding process and genetic operators are the same as that of GPCS and GPDPS. This new LS combines both tournament selection and LS. More precisely, it utilises tournament selection for the initial h generations and switches to LS afterward.

The use of the switching criterion is to tackle the hyper-selection issue

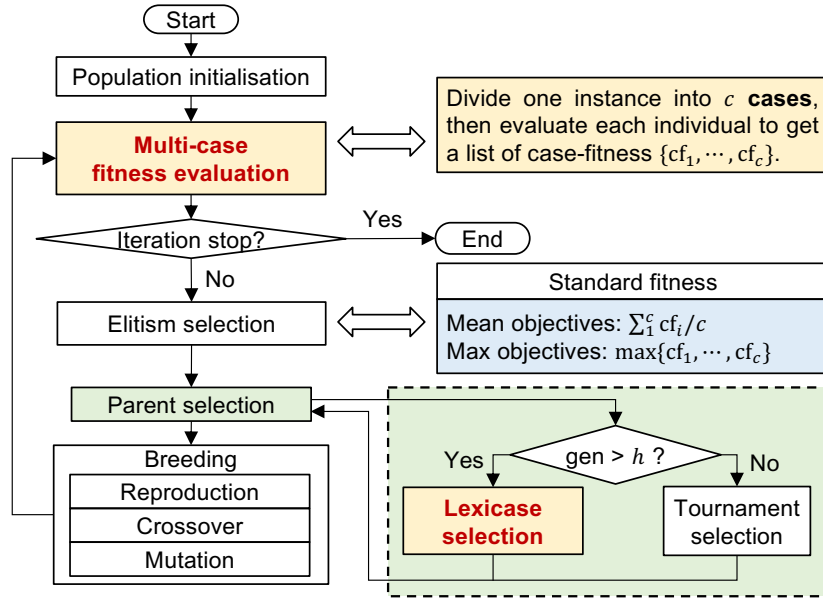


Figure 3.6: The flowchart of the proposed GPLS algorithm.

[91] of LS, i.e., if an individual performs significantly better than others on most cases, then it will be selected much more often than other individuals. This is more likely to occur at the early stage of the evolutionary process, where most of the individuals are randomly generated and the population is very diverse. In this case, it is likely that only a few elite individuals dominate most other individuals on most cases. LS tends to always select these elite individuals regardless of the order of the case, and the population diversity can be lost very fast. Therefore, we propose to use tournament selection in the first h generations, and then switch to LS when most individuals are good enough and can specialise on different cases.

The New Lexicase Selection

The new LS selects parents by tournament selection in the first h generations, and then switches to LS afterwards. The traditional tournament se-

lection is adopted, which first samples k individuals randomly, and then selects the one with the best standard fitness. More detailed information on LS is provided below.

The LS used in this chapter is extended from the ϵ -LS [123]. First, a *candidate pool* of individuals is created by randomly sampled individuals from the population. In other words, the candidate pool is a random subset of the population. Then, the traditional ϵ -LS is applied to the candidate pool to select the parent. The use of the candidate pool borrows the advantages of limiting the selection pressure [231] of LS.

Algorithm 4: Lexicase selection for DFJSS.

Input: The population: pop ; Number of cases: c ; Candidate pool size: p .

Output: The selected individual: y .

```

1 Get a random permutation  $\phi \leftarrow \text{shuffle}(1, \dots, c)$ ;
2 Create the candidate pool  $\Gamma$  by randomly selecting  $p$  individuals with no
  replacement from  $pop$ ;
3 for  $j = 1 \rightarrow c$  do
4    $a \leftarrow \phi[j]$ ;
5   Get the best case-fitness  $\mathbf{cf}_a^*$ :  $\mathbf{cf}_a^* \leftarrow \min\{\mathbf{cf}_a(x) \mid x \in \Gamma\}$ ;
6    $\epsilon \leftarrow \text{CalculateEpsilon}(\Gamma, a)$ ;
7   for  $x \in \Gamma$  do
8     if  $\mathbf{cf}_a(x) > \mathbf{cf}_a^* + \epsilon$  then Remove  $x$  from  $\Gamma$ ;
9   end
10  if there is only one individual in  $\Gamma$  then
11    return  $y \leftarrow \Gamma[1]$ ;
12  end
13 end
14 Random select an individual  $y$  from candidates  $\Gamma$ ;
15 return  $y$ ;
```

Algorithm 4 shows the LS for DFJSS. After shuffling the order of the cases (Line 1) and creating the candidate pool Γ with p individuals (Line 2), the individuals in the candidate pool are removed case by case. For each case a , the best case-fitness \mathbf{cf}_a^* among all the individuals in the candidate pool is first identified. Then, the ϵ parameter is calculated for the

current case by the function $\text{CalculateEpsilon}(\Gamma, a)$, which will be described in Algorithm 5. Then, all the individuals in the candidate pool whose case-fitness on a is worse than $\text{cf}_a^* + \epsilon$ are removed from Γ . This individual removal is repeated for all the cases in the shuffled order ϕ , or until there is only a single individual left in Γ (Line 11). If there are still multiple individuals left in Γ after enumerating all the cases, then the parent is randomly selected from Γ (Line 14).

In Algorithm 4, the ϵ parameter is the threshold to accept slightly worse individuals than the best in a case. It is important to set a proper ϵ value to be neither too greedy nor too random. Here, we set the ϵ value based on the distribution of the case fitness of the individuals in the candidate pool [180]. It is described in Algorithm 5. First, we calculate the median of the case fitness of all the individuals in Γ on case a . Then, we calculate the difference between each case-fitness and the median. Finally, we set ϵ to the median of all such differences. By using the median of all the differences, we can have a more robust estimation on the variance of the case fitness in the population than by using a fixed ϵ value or the mean, as the median is less sensitive to outliers. Such advantage is more obvious in the GP for DFJSS, where the training instance is changed at each new generation, which can result in large noise in the fitness evaluation [264].

Algorithm 5: $\text{CalculateEpsilon}(\Gamma, a)$.

Input: The candidate pool: Γ ; index of current case: a .

Output: The epsilon: ϵ

- 1 Calculate the median of the current case fitness of all the individuals in the candidate pool: $\mu \leftarrow \text{median}\{\text{cf}_a(x) \mid x \in \Gamma\}$;
// Calculate the difference between each case fitness and the median
 - 2 **for** $x \in \Gamma$ **do**
 - 3 $\delta(x) \leftarrow |\text{cf}_a(x) - \mu|$;
 - 4 **end**
 - 5 Set ϵ to the median of all the differences: $\epsilon \leftarrow \text{median}\{\delta(x) \mid x \in \Gamma\}$;
 - 6 **return** ϵ ;
-

3.3 Experimental Design

3.3.1 Dataset

In the experiment, we use the simulation model [264] that assumes 5000 jobs need to be processed by 10 heterogeneous machines. The processing rate of each machine is randomly generated within the range $[10, 15]$. The travel time between machines and the travel time between the entry/exit point and each machine is sampled from a uniform discrete distribution between 7 and 100.

For DFJSS simulation, new jobs will arrive over time according to a Poisson process with a rate λ . Each job has a different number of operations that are randomly generated by a uniform discrete distribution between 2 and 10. In addition, the importance of jobs might be different, which is indicated by weights. The weights of 20%, 60%, and 20% of jobs are set as one, two, and four, respectively. The workload of each operation is assigned by uniform discrete distribution within the range $[100, 1000]$. The due date factor is set to 1.5, which means that the due date of a job is set to 1.5 times of its total processing time after its arrival time.

The utilisation level is a key factor in simulating different DFJSS scenarios. A higher utilisation level indicates a generally busier job-shop scenario. In this chapter, we consider eight scenarios based on four objectives and two utilisation levels (0.85 and 0.95). To ensure the accuracy of the collected data, warm-up jobs (the first 1000 jobs) were used to obtain typical scenarios that occur in the long-term simulation of the DFJSS system. Then, data were collected for the next 4000 jobs. The simulation stops after completing 6000 jobs. This way, we can almost make sure that the first arrived 4000 jobs have been completed. We generate a single replication of simulation, but change the random seed at each generation of GP to improve the generalisability of the evolved scheduling heuristics [94].

Table 3.1: The description of terminals and functions.

Terminal	Description
TIS	time stay in the system = $t - \text{arrivalTime}$
W	the job weight
NOR	the remaining operation number of the job
WKR	the remaining work
rDD	the relative due date = $DD - t$
SLACK	the slack
PT	the processing time for the operation
OWT	the waiting time for the operation = $t - \text{ORT}$
NPT	the median processing time for the succeeding operation
MWT	the waiting time for the machine = $t - \text{MRT}$
NIQ	the operation number in the machine waiting queue
WIQ	the total work in the machine waiting queue
TRANT	the transportation time
Functions	$\max, \min, +, -, \times, \text{protected} /$

* t : current time; arrivalTime: job arrival time; DD: due date; ORT: ready time of operation; MRT: ready time of machine;

3.3.2 Parameter Setting

In the experiments, the terminal and function sets of GP are shown in Table 3.1. The terminal set consists of the features related to machines (e.g., NIQ, WIQ, and MWT), operations (e.g., PT, NPT, and OWT), jobs (e.g., WKR, NOR, W, and TIS), and transport (e.g., TRANT). In the function set, the arithmetic operators take two arguments. The “/” operator is protected and returns 1 if divided by zero. The *max* and *min* functions take two arguments and return the maximum and minimum of their arguments, respectively. The proposed algorithm is implemented based on an open-source platform called ECJ [138]. The GP parameter settings are shown in Table 3.2. For the parent selection, cluster selection, DPS, LS, and tournament selection are used. In the experiments, we set the minimal number of cases to 10 so that there are enough cases for the LS to play a reasonable role. The maximal number of cases is set to 100 (so that each case contains

at least 40 jobs) to control the uncertainty of the case-fitness.

Table 3.2: The parameter settings of GP.

Parameter	Value
Population size	1000
Number of generations	50
Method for initialising population	Ramped-half-and-half
Initial minimum/maximum depth	2 / 6
Elitism	10
Maximal depth	8
Crossover rate	0.80
Mutation rate	0.15
Reproduction rate	0.05
Terminal/non-terminal selection rate	10% / 90%
Switching criterion h	5
The number of cases c	10, 25, 40, 50, 80, 100
Candidate pool size p	10, 20, 30, 40, 50, 60, 70, 80
	100, 200, 400, 600, 800, 1000

3.3.3 Comparison Design

The proposed GPLS algorithm has two important parameters, which are the number of cases (c) and candidate pool size (p). At the beginning, a sensitivity analysis is done to select the best pair of parameters (c, p). Then, to verify the effectiveness of the proposed three novel parent selection mechanisms (GPCS, GPDPS, and GPLS) algorithm, we compare them with the following algorithms:

1. **GP7**: The traditional GP approach that uses tournament selection with size 7. This is the standard GP hyper-heuristic for DFJSS.
2. **GP4**: The traditional GP approach that uses tournament selection with size 4. It is known that tournament size can control the selection pressure of tournament selection [229]. By reducing the tourna-

ment size from 7 to 4, the traditional GP can have a better population diversity.

3. **GPLSⁱ**: The GPLS with multiple instances [180]. In the fitness evaluation, it uses c different instances, each of which is a short simulation with $1000/c$ warm-up jobs, and stops after completing the next $4000/c$ jobs.
4. **GPm**: GP with multi-case fitness evaluation and tournament selection with size 7. It simply sets the fitness to the mean of all the case-fitnesses. Note that GPm is also a proposed algorithm in this chapter that only uses multi-case fitness. The comparison with GPm can help find whether LS works.
5. **STS^r**: The GP with semantic tournament selection with size 7 [47]. It uses the case-fitness vector and selects the individual that is significantly better than the other based on the Wilcoxon rank sum test. The tie is broken randomly.
6. **STS^s**: The GP with semantic tournament selection with size 7 [47]. It uses the case-fitness vector, and selects the individual that is significantly better than the other based on the Wilcoxon rank sum test. The tie is broken by selecting the individual with smaller program size.

To verify different hypotheses, we form the following comparisons:

1. Comparison between GPLS and GP7, GP4, and GPLSⁱ: This can verify the effectiveness of the proposed multi-case fitness evaluation strategy.
2. Comparison between GPLS and GPm, STS^r, and STS^s: This can verify the effectiveness of the LS under the proposed multi-case fitness evaluation.

Each compared algorithm is run 30 times independently to train the scheduling heuristics for each scenario. To evaluate the test performance of the trained scheduling heuristics, we use 30 unseen simulations in the test set, each with 1000 warm-up jobs and 5000 jobs afterwards. The *test performance* is defined as the average objective value of the schedules generated by the scheduling heuristics over the 30 test simulations.

3.4 Results and Discussions

3.4.1 Sensitivity Analysis

To have a comprehensive sensitivity analysis of the c and p parameters, we consider a wide range of parameter values, i.e., $c \in \{10, 25, 40, 50, 80, 100\}$ and $p \in \{10, 20, 30, 40, 50, 60, 70, 80, 100, 200, 400, 600, 800, 1000\}$. Note that when $p = 1000$ (the entire population), the LS is equivalent to the traditional ϵ -LS.



Figure 3.7: The heat map of test performance of GPLS with different numbers of case and pool size on eight scenarios.

Figure 3.7 shows the heat map of the test performance of GPLS with different (c, p) parameter values on the 8 test scenarios. The darker (blue) colour indicates smaller values (better test performance), while the lighter (yellow) colour indicates larger values (worse test performance). From Figure 3.7, we can see that a large number of cases and small candidate pool size (bottom right region) tend to lead to poor performance, while a relatively small number of cases (e.g, 25~50) and a relatively large candidate pool size (e.g., 400~800) tend to result in promising test performance. Different scenarios show different distributions of the test performance. For example, the colors of the $\langle F_{\max}, 0.85 \rangle$ and $\langle F_{\max}, 0.95 \rangle$ scenarios are much darker than the $\langle F_{\text{mean}}, 0.85 \rangle$ and $\langle F_{\text{mean}}, 0.95 \rangle$ scenarios. However, the best regions for all the scenarios are consistently located around the top left.

In addition, we compare each parameter setting with the baseline GP7, using the Wilcoxon rank sum test with a significance level of 0.05. We found that 9 parameter settings, i.e., (10, 200), (25, 80), (25, 200), (25, 400), (25, 600), (25, 800), (40, 200), (40, 800) and (80, 600), can obtain significantly better than GP7 on 7 out of the 8 test scenarios. Among these 9 settings, (25, 800) performed slightly better than others. Therefore, we select $c = 25$ and $p = 800$ for GPLS in the subsequent experiments.

3.4.2 Test Performance

In this section, we first compare the proposed GP methods with three novel parent selection methods to baseline GP with the standard tournament selection (GP7) to find which one of the proposed parent selection methods is the best.

Table 3.3 shows the mean (standard deviation) of the test performance of the 30 independent runs of GPCS, GPDPS, and GPLS on the 8 scenarios. We have also conducted the Wilcoxon rank sum test with a significance level of 0.05 to make pairwise comparisons (each algorithm is compared

Table 3.3: The mean (standard deviation) of the test performance of the 30 independent runs of GP7, GPCS, GPDPS, GPLS on 8 scenarios.

S*	GP7	GPCS	GPDPS	GPLS
1*	1291.40(12.62)	1297.73(19.28)(=)	1284.45(12.46)(↑)(↑)	1282.77(13.77)(↑)(↑)(=)
2*	1375.33(16.77)	1380.39(19.94)(↓)	1368.16(23.17)(↑)(↑)	1359.44(13.49)(↑)(↑)(=)
3*	524.54(2.93)	524.39(3.14)(=)	525.54(4.86)(=)(=)	523.46(3.56)(↑)(↑)(↑)
4*	566.69(3.23)	568.50(4.52)(=)	566.83(3.20)(=)(=)	565.84(2.62)(=)(↑)(=)
5*	733.27(14.69)	735.60(13.58)(=)	719.52(11.28)(↑)(↑)	717.05(13.34)(↑)(↑)(=)
6*	860.42(19.09)	858.33(18.67)(=)	847.18(18.69)(↑)(↑)	831.09(11.74)(↑)(↑)(=)
7*	2323.13(161.67)	2319.98(109.23)(=)	2246.78(73.55)(↑)(=)	2240.34(65.35)(↑)(↑)(=)
8*	2460.96(100.40)	2445.11(97.36)(=)	2426.09(93.69)(=)(=)	2398.00(110.64)(↑)(=)(=)

* S: Scenarios, 1: <Fmax, 0.85>, 2: <Fmax, 0.95>, 3: <Fmean, 0.85>, 4: <Fmean, 0.95>, 5: <Tmax, 0.85>, 6: <Tmax, 0.95>, 7: <WTmax, 0.85>, 8: <WTmax, 0.95>.

with all the algorithms to its left in the table). The “↑/↓/=” after each entry in the table indicates that the corresponding results are significantly better/worse than or similar to the results of the compared algorithm. For example, for the <Fmax, 0.85> scenario, the (↑)(↑)(=) for GPLS indicates that GPLS performed significantly better than GP7 and GPCS, and there is no statistical difference between GPLS and GPDPS.

From the table, it is evident that GPLS consistently achieves significantly superior performance compared to GP7 and GPCS across 7 out of the 8 test scenarios. Furthermore, GPLS outperforms GPDPS on a specific scenario (<Fmean, 0.95>) and demonstrates comparable performance with GPDPS on all other scenarios. Notably, GPDPS exhibits notably superior performance over GP7 on 5 out of 8 scenarios and outperforms GPCS on half of the 8 scenarios. GPCS, on the other hand, exhibits notably poorer performance than GP7 on one scenario (<Fmax, 0.95>) and performs similarly to GP7 on the remaining scenarios. The superiority of GPLS and GPDPS over GP7 and GPCS verify the effectiveness of the proposed LS and DPS. However, the proposed cluster selection does not contribute significantly to improving test performance. Among the three

novel parent selection methods, LS emerges as the most effective. To further explore the components related to GPLS, we conduct two additional comparisons focusing on the effectiveness of the newly proposed multi-case fitness evaluation and LS, respectively.

Effectiveness of Multi-Case Fitness Evaluation

To verify the effectiveness of the multi-case fitness evaluation strategy, we compare GPLS with GP7, GP4, and GPLSⁱ. GP4 uses the tournament selection but has a better population diversity by reducing the tournament size from 7 to 4. GPLSⁱ uses a case-fitness list, where each case-fitness is based on a set of short instances/simulations.

Table 3.4: The mean (standard deviation) of the test performance of the 30 independent runs of GP7, GP4, GPLSⁱ, GPLS on 8 scenarios.

S*	GP7	GP4	GPLS ⁱ	GPLS
1*	1291.40(12.62)	1294.29(10.08)(=)	1280.10(10.95)(↑)(↑)	1282.77(13.77)(↑)(↑)(=)
2*	1375.33(16.77)	1378.48(16.63)(=)	1362.07(18.37)(↑)(↑)	1359.44(13.49)(↑)(↑)(=)
3*	524.54(2.93)	523.28(2.36)(↑)	523.07(2.07)(↑)(=)	523.46(3.56)(↑)(=)(=)
4*	566.69(3.23)	567.02(3.67)(=)	596.42(175.53)(=)(=)	565.84(2.62)(=)(=)(=)
5*	733.27(14.69)	740.84(16.44)(↓)	722.33(14.49)(↑)(↑)	717.05(13.34)(↑)(↑)(=)
6*	860.42(19.09)	861.83(17.92)(=)	835.51(13.01)(↑)(↑)	831.09(11.74)(↑)(↑)(=)
7*	2323.13(161.67)	2239.42(45.33)(=)	2224.77(56.65)(↑)(↑)	2240.34(65.35)(↑)(↑)(=)
8*	2460.96(100.40)	2494.17(123.86)(=)	2443.69(117.17)(=)(↑)	2398.00(110.64)(↑)(↑)(=)

* S: Scenarios, 1: <Fmax, 0.85>, 2: <Fmax, 0.95>, 3: <Fmean, 0.85>, 4: <Fmean, 0.95>, 5: <Tmax, 0.85>, 6: <Tmax, 0.95>, 7: <WTmax, 0.85>, 8: <WTmax, 0.95>.

Table 3.4 shows the mean (standard deviation) of the test performance of the 30 independent runs of GP7, GP4, GPLSⁱ, and GPLS on the 8 scenarios. From the table, we can see that GPLS obtains significantly better performance than GP7 and GP4 on 7 out of the 8 test scenarios. GPLS performs similarly to GPLSⁱ on all the scenarios. In addition, GPLSⁱ performed significantly better than GP7 and GP4 on 6 out of the 8 test scenarios, which is slightly worse than GPLS. The advantages of GPLS and

GPLSⁱ over GP7 and GP4 verify the effectiveness of the use of multi-case fitness instead of a single aggregated fitness in parent selection for evolving DFJSS scheduling heuristics.

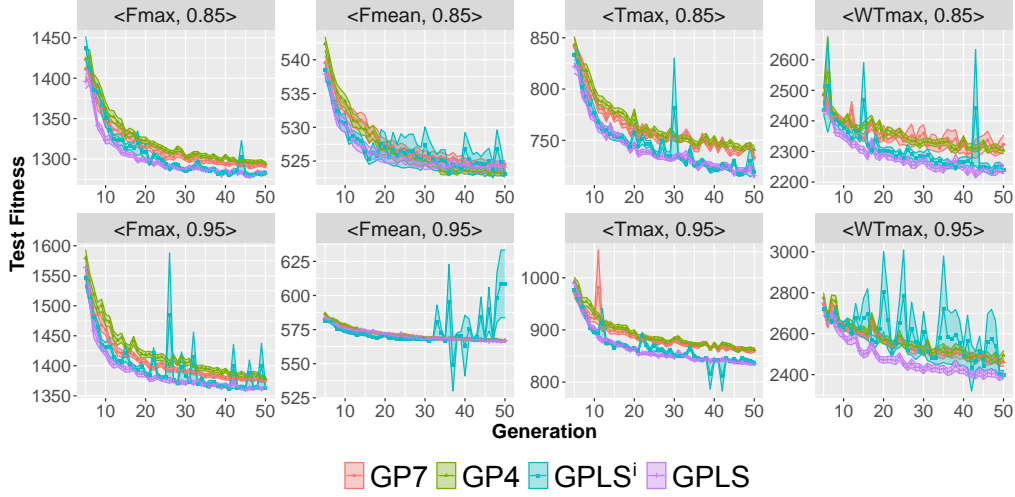


Figure 3.8: The convergence curves of the test performance of the compared methods on the 8 scenarios.

Figure 3.8 shows the convergence curves of the test performance of GP7, GP4, GPLSⁱ, and GPLS on the 8 scenarios. From the figure, we can see that GPLS has a faster convergence than GP7 and GP4, as its convergence curves are almost always below that of GP7 and GP4 for all the scenarios. This verifies the effectiveness of the proposed multi-case fitness selection strategy. On the other hand, although GPLSⁱ shows competitive final test performance in Table 3.4, its convergence curves in Figure 3.8 are shown to be quite fluctuating and unstable, especially in the <Fmean, 0.95> scenario, where it shows worse test performance and large variance. A possible reason is that GPLSⁱ uses several small-scale instances (160 jobs and 40 warm-up jobs) for evaluation, and these instances are all from the beginning stage of the scheduling system and the small number of warm-up jobs can not simulate a stable scheduling system very well. This leads to overfitting and worse generalisation of the evolved scheduling heuris-

tics.

Overall, the advantages of GPLS over GP7, GP4, and GPLSⁱ verify the effectiveness of the proposed multi-case fitness evaluation.

Effectiveness of Lexicase Selection

To verify the effectiveness of the use of LS, we compare GPLS with GPm, STS^r, and STS^s. All the compared algorithms employ the multi-case fitness, but have different parent selection schemes. GPm simply averages the case-fitnesses into a single fitness, and uses the tournament selection. For the mean-objectives, it becomes equivalent to the baseline GP. However, for the max-objectives, its fitness is different from the standard fitness. STS^r and STS^s select parents by conducting statistical significance test on the case-fitness lists. They are also state-of-the-art GP parent selection methods that improve population diversity.

Table 3.5: The mean (standard deviation) of the test performance of the 30 independent runs of GPm, STS^r, STS^s, GPLS on 8 scenarios.

S*	GPm	STS ^r	STS ^s	GPLS
1*	1282.42(12.36)	1317.59(19.20)(↓)	1372.22(65.59)(↓)(↓)	1282.77(13.77)(=)(↑)(↑)
2*	1367.34(19.42)	1406.85(22.18)(↓)	1488.40(81.23)(↓)(↓)	1359.44(13.49)(↑)(↑)(↑)
3*	523.87(3.38)	526.20(4.51)(↓)	532.30(6.40)(↓)(↓)	523.46(3.56)(=)(↑)(↑)
4*	567.73(3.27)	570.20(5.02)(↓)	578.63(3.84)(↓)(↓)	565.84(2.62)(↑)(↑)(↑)
5*	720.74(15.91)	763.96(19.89)(↓)	835.54(46.86)(↓)(↓)	717.05(13.34)(=)(↑)(↑)
6*	830.88(11.50)	883.68(23.94)(↓)	971.03(59.00)(↓)(↓)	831.09(11.74)(=)(↑)(↑)
7*	2250.66(64.28)	2358.00(64.15)(↓)	2464.33(98.52)(↓)(↓)	2240.34(65.35)(=)(↑)(↑)
8*	2428.43(79.14)	2586.20(117.66)(↓)	3434.35(412.91)(↓)(↓)	2398.00(110.64)(=)(↑)(↑)

* S: Scenarios, 1: <Fmax, 0.85>, 2: <Fmax, 0.95>, 3: <Fmean, 0.85>, 4: <Fmean, 0.95>, 5: <Tmax, 0.85>, 6: <Tmax, 0.95>, 7: <WTmax, 0.85>, 8: <WTmax, 0.95>.

Table 3.5 shows the mean (standard deviation) of the test performance of the 30 independent runs of GPm, STS^r, STS^s, and GPLS on the 8 scenarios. From the table, it can be seen that GPLS achieved significantly better performance than GPm on 2 scenarios, and is comparable with GPm on

all the remaining scenarios. It significantly outperformed STS^r and STS^s for all the 8 scenarios. This verifies the effectiveness of the proposed LS.

GPm obtained significantly better performance than GP7 on 5 scenarios, which shows that even averaging the case-fitness can improve the performance, especially on the max-objectives. This is because the multi-case fitness evaluation uses more information, e.g., the completion time of c jobs (each for a case) rather than a single job with the maximal objective value. This can improve the generalisation. This might also be one of the reasons that GPLS performs well. However, STS^r and STS^s perform significantly worse than GP7 on almost all the 8 scenarios. This is contradictory to the results in [47], where STS^r and STS^s performed better than tournament selection and semantic in selection [70] for solving regression problems. A possible reason is that the DFJSS problem is more complex than regression problems and the strategy to select parents based on the statistical test might give poor individuals high probabilities to be selected in DFJSS.

In summary, the above two comparisons with different sets of other existing GP approaches verify the effectiveness of the proposed GPLS, and the newly proposed multi-case fitness evaluation and LS.

3.4.3 Distribution of Parent Fitness

Using the multi-case fitness evaluation instead of the standard fitness is expected to reduce the selection pressure and allow more specialist individuals with worse standard fitness to be selected as parents.

To analyse such behaviour of the multi-case fitness evaluation, we plot the distribution of the standard fitness of the selected parents in a run of GP7, GPm, and GPLS. Figure 3.9 shows the distributions at generation 5, 20, 35, and 45 in the $\langle F_{\max}, 0.85 \rangle$ scenario. Note that all three algorithms use the tournament selection on the standard fitness until generation 5. Thus, They have exactly the same process in the first generation, and the

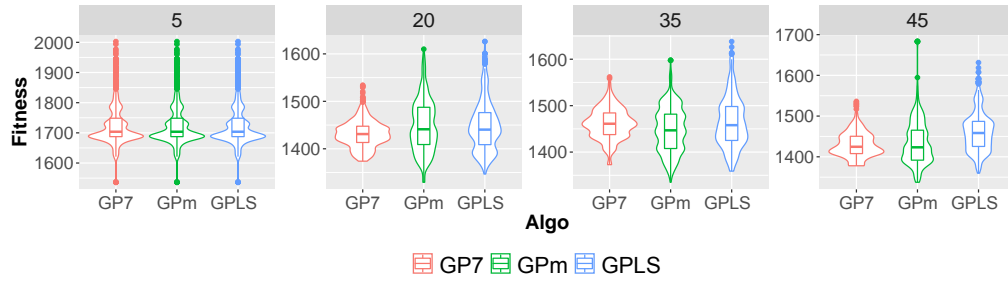


Figure 3.9: The violin plots of the standard fitness of the selected parents of GP7, GPM, and GPLS at generation 5, 20, 35, and 45 in scenarios $\langle F_{\max}, 0.85 \rangle$ of a single run.

same distribution of parent fitness at generation 5. After that, we can see that both GPM and GPLS achieved a wider-spread distribution than GP7. An interesting pattern is that GPM and GPLS were able to select parents with better standard fitness than GP7. This indicates that the multi-case fitness evaluation can help the algorithms generate better individuals, which can then be used as parents in subsequent generations.

3.4.4 Phenotypic Diversity

Besides the fitness, the phenotypic characterisation (PC) [93] is a more comprehensive representation of the behaviour of a GP rule. The PC of a GP rule is represented as a numeric vector, where each dimension is the decision made by the rule under a decision situation (e.g., the index of the selected candidate by an idle machine from its queue).

Figure 3.10 shows the convergence curves of the number of unique PCs in the population in GP7, GP4, and GPLS on the 8 scenarios. From the figure, we can see that GP4 always contains more unique PCs in the population than GP7 for all the scenarios, which is expected. The convergence curves of GPLS and GP7 overlap with each other in the first 5 generations due to the switching criterion. After generation 5, GPLS starts to have more unique PCs than GP7. Compared to GP4, GPLS has more unique

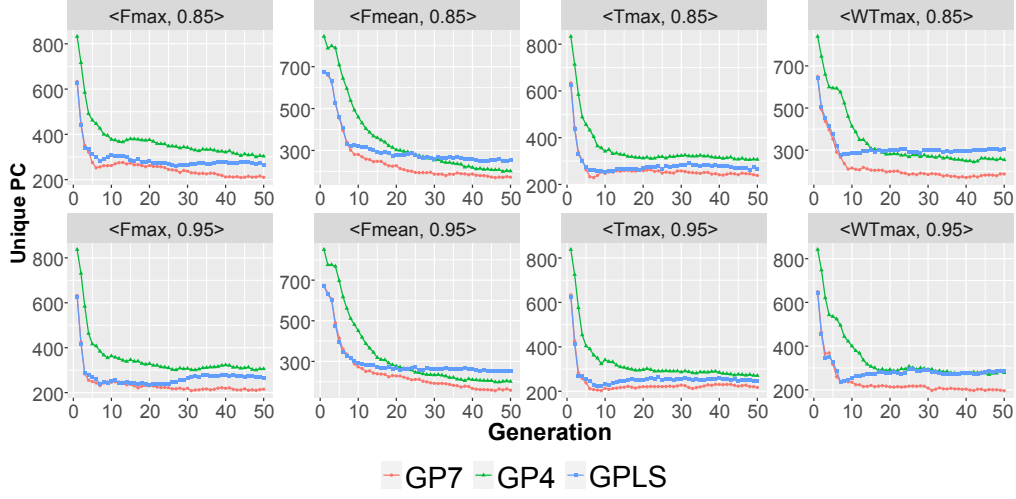


Figure 3.10: The convergence curves of unique PCs of GPLS and comparison methods on eight scenarios.

PCs on three scenarios ($\langle \text{Fmean}, 0.85 \rangle$, $\langle \text{Fmean}, 0.95 \rangle$, $\langle \text{WTmax}, 0.85 \rangle$) in the later stages of the evolutionary process, while has fewer unique PCs on four scenarios ($\langle \text{Fmax}, 0.85 \rangle$, $\langle \text{Fmax}, 0.95 \rangle$, $\langle \text{Tmax}, 0.85 \rangle$, $\langle \text{Tmax}, 0.95 \rangle$). Considering Table 3.4, where GPLS significantly outperformed GP4 although it has slightly lower phenotypic diversity, we can see that increasing phenotypic diversity, although important, is not the only reason for the performance improvement in GPLS.

3.5 Further Analyses

3.5.1 Correlation between Case-Fitnesses

For LS based on the case-fitness list, an important factor is the correlation between the fitnesses of different cases. Ideally, the population should contain individuals that specialise at different cases to increase the selection diversity. If the case-fitnesses are highly correlated, the selected parent will be the same regardless of the order of the case. That is, if an indi-

vidual A is better than B on one case, it is highly likely to be better on other cases. As a result, LS has no advantage over the tournament selection.

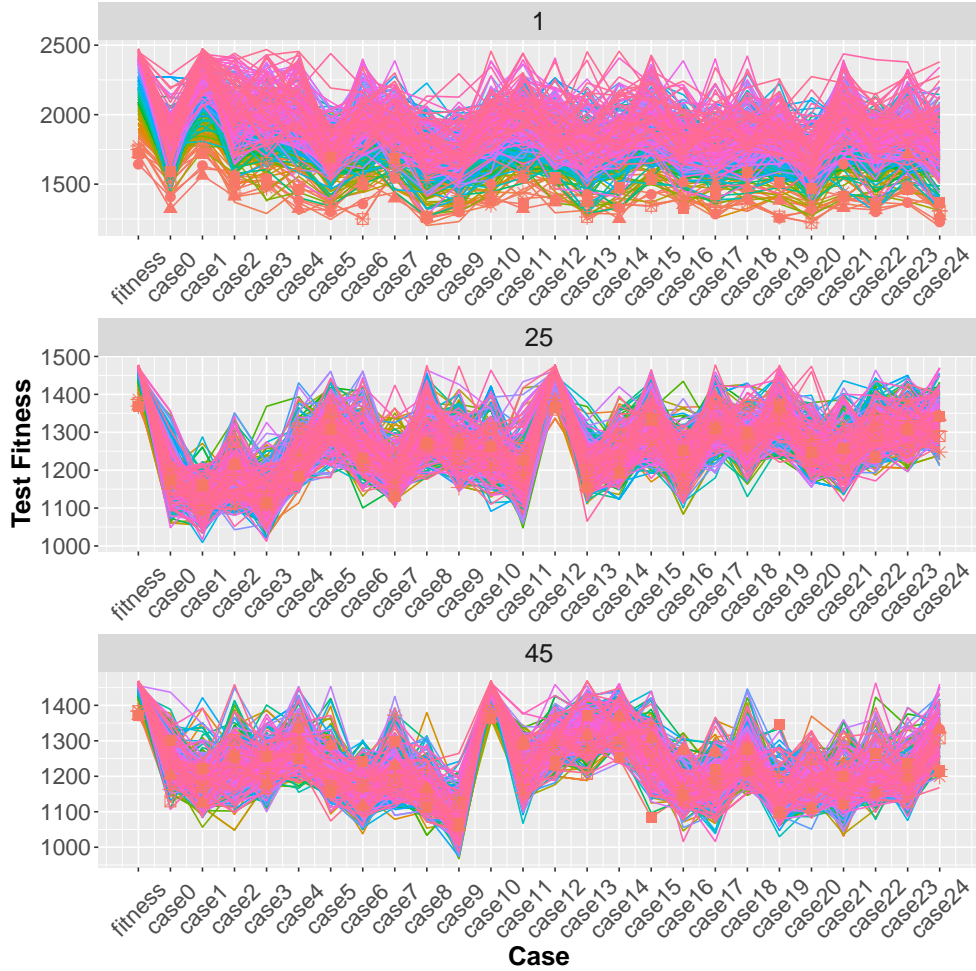


Figure 3.11: The standard fitness and case-fitness of the top 400 individuals of GPLS at generation 1, 25 and 45 on the $\langle F_{\text{mean}}, 0.85 \rangle$.

To investigate the correlation between case-fitnesses, we plot the case-fitnesses of the top 400 individuals in the population of GPLS (that are most likely to be selected by LS) in the early stage (generation 1), middle stage (generation 25) and late stage (generation 45) for the $\langle F_{\text{max}}, 0.85 \rangle$ and $\langle F_{\text{mean}}, 0.85 \rangle$ scenarios. Figures 3.11 and 3.12 show the re-

sults, where each individual corresponds to a line and the standard fitness of the individual is also given on the most left as a reference.

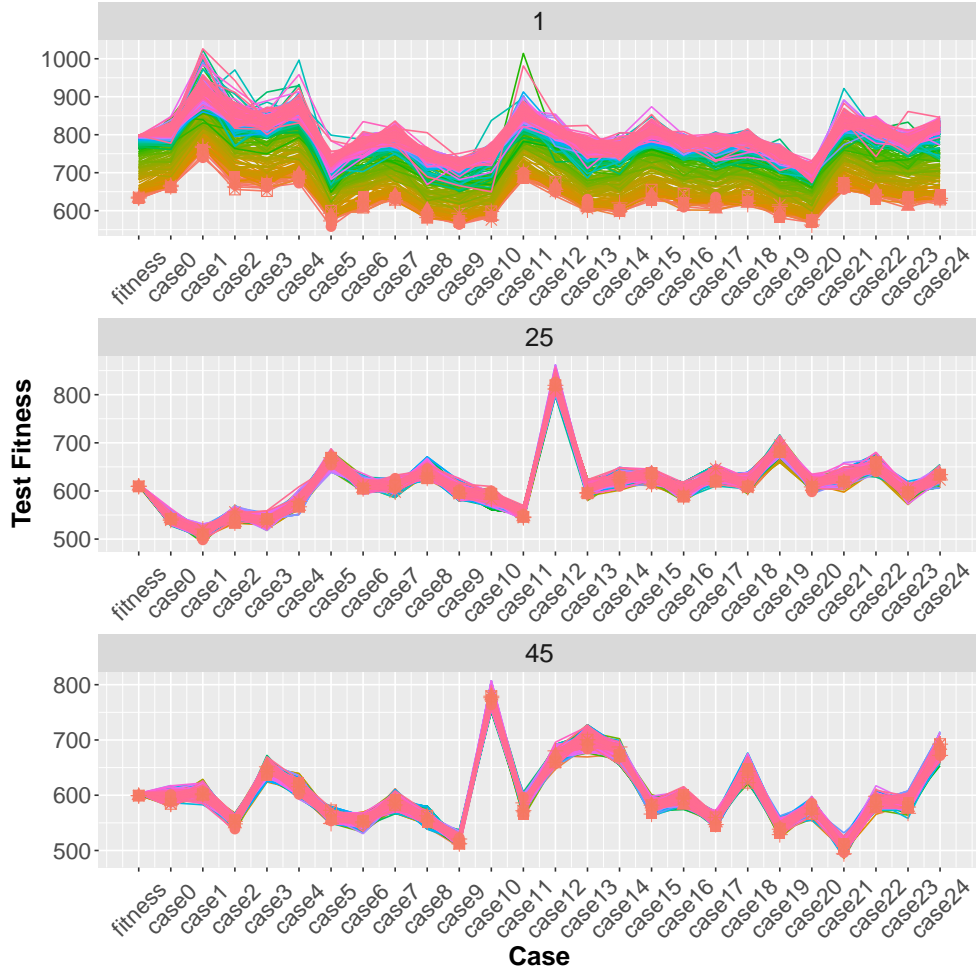


Figure 3.12: The standard fitness and case-fitness of the top 400 individuals of GPLS at generation 1, 25, and 45 on the $\langle F_{\max}, 0.85 \rangle$.

From the figures, we can see that in both scenarios, initially there are a few individuals that perform well on almost all the cases (their lines are below the others). However, as the evolution proceeds, the lines of different individuals become more mixed together, and it is difficult to see any individual's line always below that of the others for all the cases. This

suggests that we can select different individuals under different orders of cases. In addition, we see that the distribution of the case-fitness has a higher variance in the $\langle F_{\max}, 0.85 \rangle$ scenario than that in the $\langle F_{\text{mean}}, 0.85 \rangle$ scenario. This is expected since the F_{\max} objective is more sensitive to outliers (the job with the maximal flowtime).

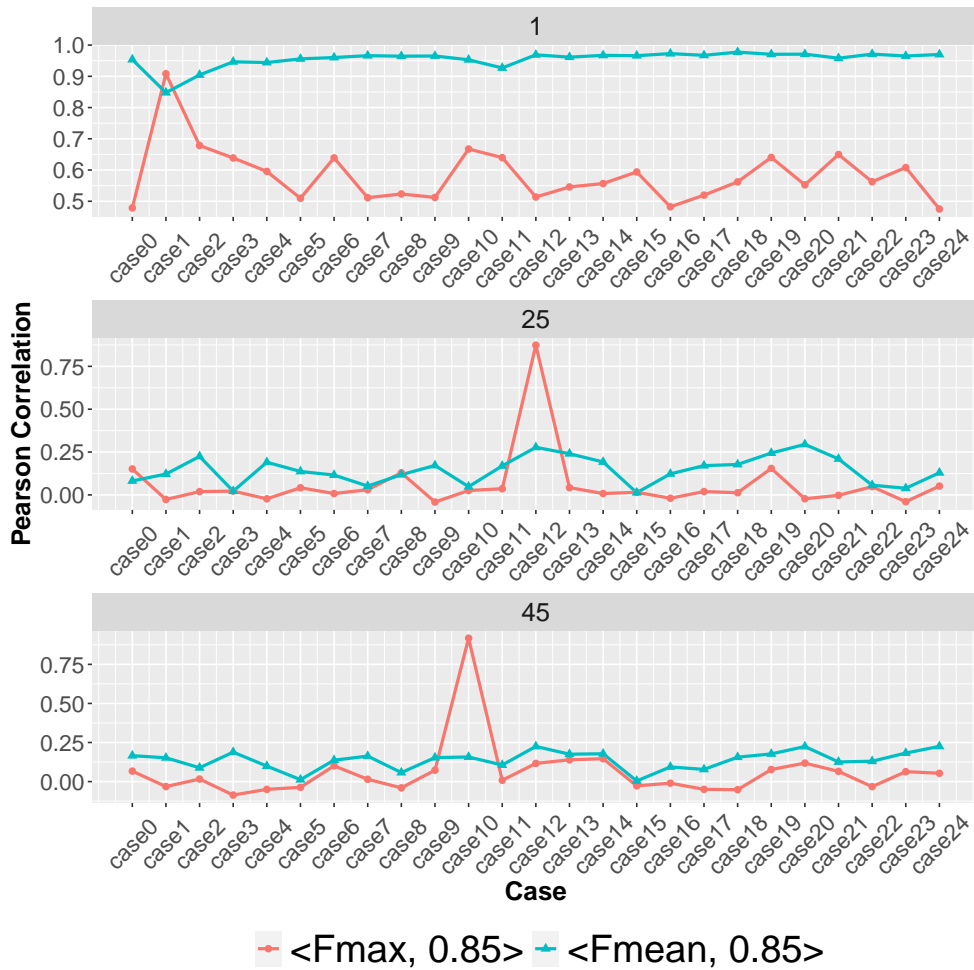


Figure 3.13: The Pearson correlation between standard fitness and case-fitness of the top 400 individuals of GPLS at generation 1, 25 and 45 on the $\langle F_{\max}, 0.85 \rangle$ and $\langle F_{\text{mean}}, 0.85 \rangle$.

We also calculate the Pearson correlation between the standard fitness

and each case-fitness of the top 400 individuals at generations 1, 25, and 45, as shown in Figure 3.13. We can see that in generation 1, all the case-fitnesses are very highly correlated with the standard fitness. This is consistent with our observations in Figures 3.11 and 3.12. However, generations 25 and 45 show different patterns. Specifically, in the $\langle F_{\max}, 0.85 \rangle$ scenario, all the cases except case 11 in generation 25 and case 9 in generation 45 have low correlations with the standard fitness. This implies that the standard fitness is mostly determined by a single case, which contains the job with the largest flowtime. In the $\langle F_{\text{mean}}, 0.85 \rangle$ scenario, all the cases have low correlations with the standard fitness. This suggests that we can select a wide range of specialist individuals by LS.

3.5.2 Frequency of Case Usage

Different cases might have different abilities to distinguish promising individuals. For example, if a case usually leads to ties between individuals, then other subsequent cases are needed to further identify the best individual. In this sense, different cases might be used at different frequencies.

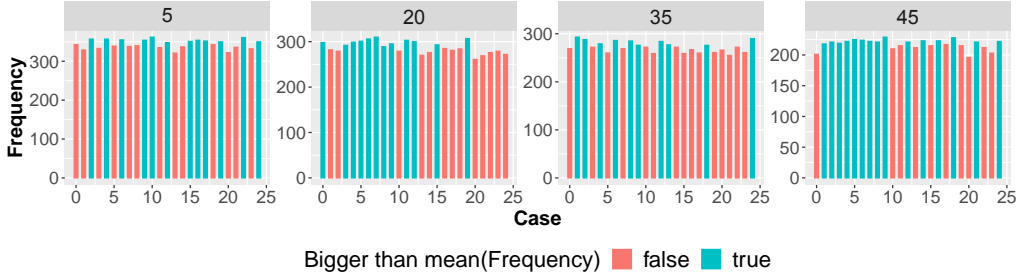


Figure 3.14: The frequency of the cases used by GPLS at generations 5, 20, 35, and 45 in the $\langle F_{\max}, 0.85 \rangle$ scenario.

To investigate this issue, we plot the frequency that each case is used by GPLS at generations 5, 20, 35, and 45 in the $\langle F_{\max}, 0.85 \rangle$ scenario, as shown in Figure 3.14. It can be seen that different cases have very similar

frequencies to each other. This indicates that all the cases have similar abilities to distinguish the individuals, which is a good sign.

3.5.3 Rule Size

In this section, we investigate how the proposed multi-case fitness evaluation strategy and LS affect the size of the evolved rules by comparing GPLS with GP7 (baseline GP). Tables 3.6 and 3.7 show the mean and standard deviation of the sizes of the evolved best routing rules and sequencing rules of GPLS and GP7 (baseline GP) on the 8 scenarios. In the tables, the “+/-/=” after each entry indicates that the corresponding rule size obtained by GPLS is significantly larger/smaller than or similar to the size obtained by GP7.

Table 3.6: The mean (standard deviation) of the size of the evolved best routing rules of 30 independent runs of GP7 and GPL for the 8 scenarios.

Scenario	GP7	GPLS
<Fmax, 0.85>	36.53(11.65)	52.60(17.51)(+)
<Fmax, 0.95>	37.87(14.51)	53.33(16.67)(+)
<Fmean, 0.85>	52.27(15.39)	55.73(18.48)(=)
<Fmean, 0.95>	50.47(16.58)	48.33(12.14)(=)
<Tmax, 0.85>	37.27(13.31)	46.60(13.78)(+)
<Tmax, 0.95>	33.07(17.33)	44.40(14.88)(+)
<WTmax, 0.85>	40.80(14.03)	46.00(12.37)(=)
<WTmax, 0.95>	39.60(20.47)	47.33(10.35)(+)

From Table 3.6, we can see that GPLS obtained significantly larger routing rule size than GP7 on 5 scenarios (<Fmax, 0.85>, <Fmax, 0.95>, <Tmax, 0.85>, <Tmax, 0.95>, <WTmax, 0.95>) and there is no statistical significant difference on the other scenarios. Table 3.7 shows that GPLS obtained significantly larger sequencing rule size than GP on 2 scenarios (<WTmax, 0.85>, <WTmax, 0.95>) and similar on the other scenarios.

Table 3.7: The mean (standard deviation) of the size of the evolved best sequencing rules of 30 independent runs of GP7 and GPLS for the 8 scenarios.

Scenario	GP7	GPLS
<Fmax, 0.85>	42.60(15.86)	44.73(14.48)(=)
<Fmax, 0.95>	44.00(14.48)	48.47(11.86)(=)
<Fmean, 0.85>	33.67(12.14)	32.27(17.88)(=)
<Fmean, 0.95>	39.73(17.82)	29.33(13.78)(=)
<Tmax, 0.85>	48.40(17.89)	50.80(16.96)(=)
<Tmax, 0.95>	41.53(14.75)	50.67(15.08)(=)
<WTmax, 0.85>	26.27(22.64)	40.87(19.90)(+)
<WTmax, 0.95>	28.40(12.53)	45.87(19.73)(+)

The above results suggest that GPLS tends to allow larger GP rules (especially routing rules) to survive, which might be a reason of the superior performance of GPLS.

3.5.4 Insight on the Evolved Scheduling Heuristics

Figures 3.15 and 3.16 give the tree structures of the routing rule and the sequencing rule from a scheduling heuristic evolved by GPLS on the <Fmax, 0.85> scenario.

The routing rule is a combination of six terminals (TRANT, MWT, PT, NOR, WIQ, and TIS), where TRANT is the most frequently used terminal (used 5 times). It is followed by MWT and PT, which are used 2 times. NOR, WIQ, and TIS, on the other hand, are used only once. The routing rule (Figure 3.15) can be simplified to R_0 as shown in Eq. (3.8).

$$R_0 = \max\{\max\{WIQ + PT, \min\{\min\{TIS, TRANT - MWT\}, TRANT - MWT + NOR\}\} + PT, TRANT\} + TRANT \quad (3.8)$$

If the operation has been waiting for a long time (a large TIS), this rule can be further simplified as $R_1 \approx \max\{TRANT + WIQ + 2PT, 2TRANT - MWT + PT, 2TRANT\}$. For a candidate machine with an empty queue

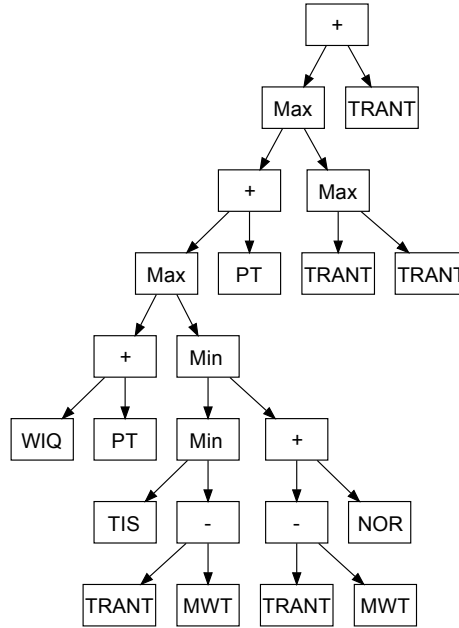


Figure 3.15: An example of the evolved routing rule by GPLS on scenario $\langle F_{\max}, 0.85 \rangle$.

($WIQ=MWT=0$), R_1 can be further simplified as $R_2 \approx \max\{TRANT + 2PT, 2TRANT + PT\}$, which means that an idle machine with a smaller transportation time (TRANT) and a smaller processing time (PT) is preferred. For a candidate machine with a busy queue (WIQ is large) of the machine (large WIQ), R_1 can be further simplified as $R_3 \approx TRANT + WIQ + 2PT$, which prefers small transportation time, small work in the queue, and small processing time. In other words, this routing rule tends to select machines with smaller transportation time (TRANT), smaller processing time (PT), and smaller work remaining in the waiting queue (WIQ).

The sequencing rule is a combination of seven terminals (WKR, TIS, WIQ, PT, NOR, W, and NPT). WKR and TIS are the most frequently used terminals, which are both used 4 times. It is followed by WIQ, PT, NOR, and W, which are used 2 times. NPT is used only once. The sequencing

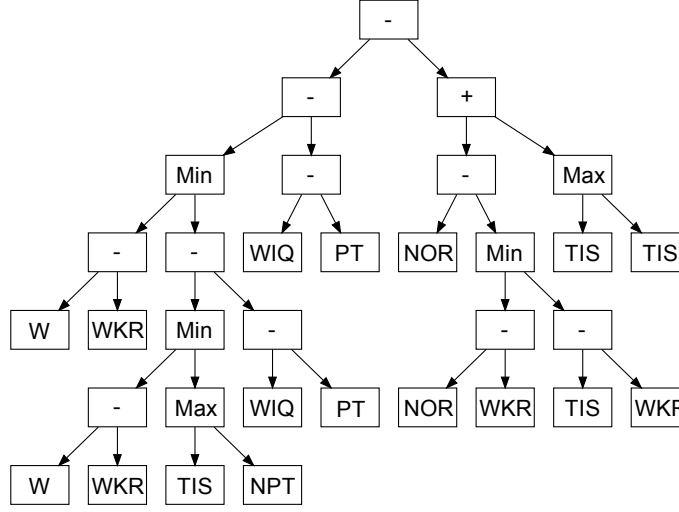


Figure 3.16: An example of the evolved sequencing rule by GPLS on scenario $\langle F_{\max}, 0.85 \rangle$.

rule (Figure 3.16) can be simplified to S_0 as shown in Eq. (3.9).

$$\begin{aligned}
 S_0 = & \min\{W - WKR, \min\{W - WKR, \max\{TIS, NPT\}\} \\
 & - (WIQ - PT)\} - (WIQ - PT) - NOR + \min\{NOR \\
 & - WKR, TIS - WKR\} - \max\{TIS, TIS\}
 \end{aligned} \tag{3.9}$$

This rule shows that the selection strategy is mainly based on the weight of the job (W), the processing time of the operation (PT), the work remaining of the operation (WKR), and the time that the operation has been waiting (TIS). Note that in most practical decision situations, the subtree $W - WKR$ outputs a smaller value than the subtree $\max\{TIS, NPT\}$, the number of operations remaining (NOR) is always smaller than TIS , and the work remaining in the waiting queue (WIQ) is always the same. Therefore, this sequencing rule can be further simplified as $S_1 \approx W + 2PT - 2WKR - TIS$. In other words, this rule tends to select operations with a smaller weight (W), a smaller processing time (PT), a larger work remaining (WKR), and a larger time in system (TIS).

Overall, we can see that for the routing decisions, the attributes of the machines, such as the transportation time and processing time (related to

the processing rate of machines) play decisive roles. For the sequencing decisions, the information of operations, such as the processing time (related to the workload of operations), time in the system, and work remaining, play decisive roles. This phenomenon is consistent with our intuition that for the max-flowtime objective, the machines that are not busy and have short transportation time are good choices to process ready operations, and the operations that stay in the system for a long time and have much remaining work should be completed as soon as possible.

3.5.5 Further Discussions

In [202], it is argued that the LS is good for solving modal problems, which is defined as follows: *“a problem is modal if a solution has to do something qualitatively different in different circumstances, that is, on inputs from different classes.”* The symbolic regression problems are proven to be modal problems [202].

DFJSS is more complex than symbolic regression, with dynamic job arrival events and complex system states. In order to improve training efficiency, we divide one instance of DFJSS into multiple cases to obtain a list of case-fitnesses, which means that different cases might not be independent. If this occurs, then the scheduling heuristics will not be qualitatively different in different circumstances (cases). Our analysis on the parents' fitness shows that the scheduling heuristics do give different scheduling solutions in different circumstances (cases), and the proposed GPLS works on the DFJSS problem. Therefore, we conservatively hypothesise that DFJSS can be considered as a modal problem, but more experiments and analyses should be performed to verify it in the future. In addition, considering the analysis of the correlation between case-fitnesses, GPLS performs significantly better, especially on max-objectives. This is because the scheduling heuristics perform more differently on max-objectives than on mean-objectives, which is consistent with the description in [202]. The

reason for this difference is that the mean-objectives focus on achieving an average performance across multiple jobs, allowing scheduling heuristics with well-rounded performance to be more competitive compared to max-objectives.

3.6 Chapter Summary

The goal of this chapter is to propose innovative diversity-based parent selection mechanisms for GP. These mechanisms aim to choose diverse individuals as parents, facilitating the generation of high-quality offspring and the evolution of effective scheduling heuristics for solving large-scale DFJSS problems involving thousands of jobs. To achieve this goal, three novel parent selection mechanisms are proposed: cluster selection, the new DPS, and the new LS.

The cluster selection method categorises individuals in the population into different clusters based on their behavioural similarity. When two parents are required for crossover, individuals are selected from different groups based on their fitness to ensure both effectiveness and diversity. The new DPS and new LS mechanisms focus on selecting individuals with expertise in different cases as parents to generate offspring. To facilitate DPS and LS, a multi-case fitness evaluation strategy is designed, efficiently dividing a large DFJSS simulation (instance) into multiple cases and extracting case-fitnesses. This strategy is more efficient than directly evaluating individuals on multiple instances. The underlying idea is that parents specialising in diverse cases are expected to contribute to the production of high-quality offspring.

The effectiveness of the proposed GP methods, incorporating the three novel parent selection mechanisms, is validated through comparisons with various existing GP approaches across eight large-scale DFJSS scenarios. Among the three mechanisms, the new LS exhibits superior performance, followed by the new DPS, with cluster selection demonstrating

relatively lower effectiveness. Further analysis shows that the effectiveness of the GPLS is attributed to the utilisation of the information of more jobs, the increased phenotypic diversity of the population by using parents with a wider fitness distribution to generate offspring, and the increased rule size.

This chapter focuses on developing effective GP methods with novel parent selection mechanisms to evolve single scheduling heuristics for solving the DFJSS problem. In the next chapter, we will study a joint decision-making mechanism by incorporating the ensemble technique into GP to evolve a group of scheduling heuristics that collaboratively make decisions to address the DFJSS problem. Specifically, the LS technique is employed as a parent selection mechanism in this chapter. In the next chapter, LS will be used to select diverse individuals, forming an ensemble that contributes to enhanced decision-making capabilities. Notably, the proposed multi-case fitness evaluation strategy, proposed in this chapter, will also be used in the next chapter to support the application of LS.

Chapter 4

Joint Decision-making by Ensemble

This chapter focuses on leveraging the ensemble technique to enhance the effectiveness of GP in evolving a group of scheduling heuristics for making joint decisions for DFJSS.

4.1 Introduction

Most of the existing GP methods for DFJSS mainly focus on evolving one scheduling heuristic [236, 253]. Recently, there has been a growing trend to learn a group of scheduling heuristics and leverage this group to make joint decisions, allowing for further exploration of the search space and the discovery of high-quality solutions [213]. Ensemble GP (EGP) is a variant of GP, incorporating GP with ensemble learning, which is able to combine multiple scheduling heuristics into an ensemble to make a decision [118]. In EGP, each scheduling heuristic in an ensemble is expected to be different and complementary to others [53]. The output of each scheduling heuristic in an ensemble is then combined using an aggregation function, such as voting or averaging, to make a final joint decision [154]. The goal of EGP is to improve the performance and generalisation ability of GP by

leveraging the strengths of multiple scheduling heuristics while mitigating their weaknesses [175]. In this way, the probability of making a sub-optimal decision in a given decision point is reduced, as the elements can complement each other to avoid poor decisions. Consequently, an ensemble is expected to be more stable than a single scheduling heuristic, which can also provide confidence and trust for users, especially in real-world applications [224]. EGP has been applied in various domains, such as regression [273] and classification [23, 154, 199], and has shown promising results in terms of performance and generalisation ability.

Recently, EGP has been applied to evolve a group of scheduling heuristics for solving scheduling problems [74, 75, 78, 211]. However, existing studies on EGP for scheduling problems are still in their early stages and exhibit several limitations. Some studies only consider the cooperation of scheduling heuristics during evaluation, while neglect the cooperation during evolution. Others treat the evolutionary processes of single individuals and ensembles independently, overlooking the direct contributions of individuals and ensembles to each other. Overall, the existing research on EGP for solving scheduling problems is still in its infancy and requires further exploration. A novel EGP method is required that has the potential to identify the strengths of both single individuals and ensembles, and make them mutually reinforcing. On the other hand, the lexicase selection is a very effective technique to improve population diversity, having the principle to evolve mutual complementary expert individuals that are good for handling different cases [240]. This highly motivates us to incorporate lexicase selection and ensemble techniques so that diverse individuals can be found by lexicase selection to form an ensemble.

4.1.1 Chapter Goals

The goal of this chapter is to *propose a novel EGP method to exploit the strengths of both single individuals and ensembles to effectively solve the DFJSS*

problems. The proposed method contains a new ensemble construction and selection strategy which is used to select diverse individuals to form ensembles and considers the evolution of single individuals and ensembles together *within a single population*. Single individuals, being simpler solutions, can efficiently explore the search space, identify promising regions, and provide useful building blocks. Ensembles, combining different individuals, might be able to get better performance by using these useful building blocks together. Ensembles with high performance can be “deconstructed” to identify the single individuals that contribute to their success. This knowledge can be used to create new, potentially even better ensembles in future generations. This creates a feedback loop where exploration by single individuals informs the creation of effective ensembles, which in turn, might discover new areas for exploration. Specifically, the objectives of this chapter are as follows:

1. Propose a new EGP method for DFJSS, called EGP^e , which enables the evolution of a population consisting of both single individuals and ensembles, offering more flexibility in the evolutionary process and better exploration of search space.
2. Design a new ensemble construction and selection strategy to help the proposed EGP^e method select diverse and high-quality individuals into an ensemble. The strategy uses lexicase selection to choose diverse individuals good at solving different cases as candidates and then employs a new similarity-checking technique to further enhance diversity.
3. Develop new crossover and mutation operators to facilitate the evolution between single individuals and ensembles in EGP^e . The offspring generated by these operators can be either single individuals or ensembles, resulting in more flexible breeding between single individuals and ensembles.

4. Analyse the effectiveness of the proposed algorithm in terms of the quality of evolved ensembles of scheduling heuristics.
5. Analyse the differences and similarities of the scheduling heuristics in the evolved ensembles by the proposed method.

4.1.2 Chapter Organisation

The rest of this chapter is organised as follows. A detailed description of the proposed algorithm is presented in Section 4.2. The experimental design is given in Section 4.3, followed by the presentation of results and discussion in Section 4.4. Additional analyses are conducted in Section 4.5. Finally, Section 4.6 concludes this chapter.

4.2 Proposed Algorithm

4.2.1 Overall Framework

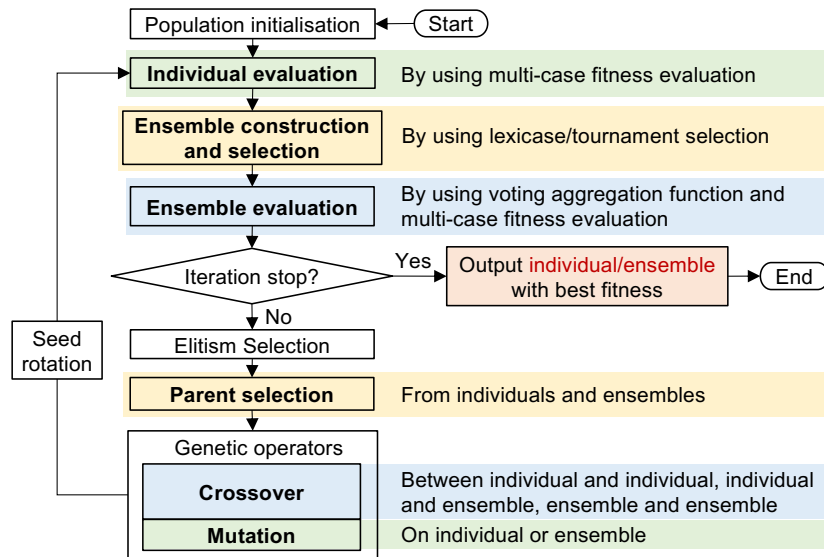


Figure 4.1: The flowchart of the proposed EGP^e method.

The flowchart of the proposed EGP^e method is shown in Figure 4.1. Population initialisation, fitness evaluation, parent selection, and breeding (crossover and mutation) are the main processes in GP, whereas special designs are developed for each part. In addition, the ensemble construction and selection strategy as well as the ensemble evaluation are newly proposed. The detailed population initialisation and the potential combinations of parents for the proposed crossover and mutation operators can be seen in Figure 4.2. It illustrates that the population comprises both single individuals and ensembles, and it presents three possible combinations of parents for crossover and two possibilities of parents for mutation. Overall, there are six differences between the proposed EGP^e and classical GP methods:

1. *Population initialisation*: EGP^e initialises and maintains a population containing both single individuals and ensembles instead of only individuals (Section 4.2.2);
2. *Individual evaluation*: EGP^e uses a multi-case fitness evaluation to assign each individual a list of case-fitnesses and standard fitness rather than only one standard fitness (Section 4.2.3);
3. *Ensemble Construction and Selection*: EGP^e uses lexicase selection to select individuals to form an ensemble, which is expected to select diverse individuals that are good at handling different cases. Additionally, a newly proposed similarity-checking strategy is used to further ensure the diversity and complementarity of elements in the ensemble (Section 4.2.4);
4. *Ensemble evaluation*: EGP^e uses a multi-case fitness evaluation to assign each ensemble a list of case-fitnesses and standard fitness (Section 4.2.5);
5. *Genetic operators*: EGP^e proposes new crossover and mutation operators to generate offspring which allows both individual(s) and en-

semble(s) to be parent(s). Also, the generated offspring can be either individual(s) or ensemble(s) (Section 4.2.6);

6. *Output*: EGP^e allows an individual or ensemble as the final output, depending on which one has the best training performance, while classical GP methods output an individual as the final result and traditional EGP methods output an ensemble as the final result. In other words, the output of EGP^e is more flexible.

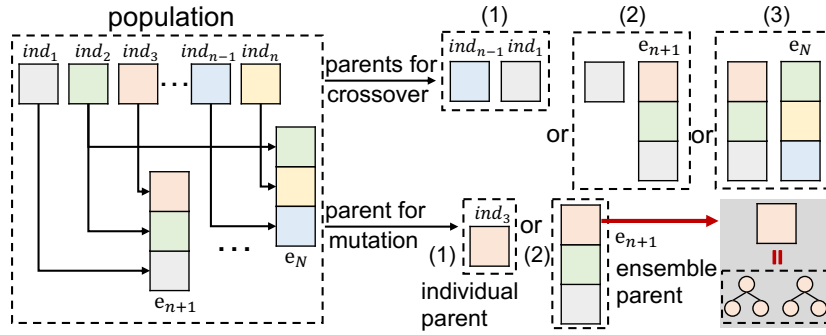


Figure 4.2: The population composition and possible parent combination(s) for crossover and mutation of the proposed EGP^e method.

4.2.2 Population Initialisation

Different from classical GP methods which hold a population of individuals only, our method initialises and maintains a population \mathbf{P} of both single individuals $\Phi = \{ind_1, \dots, ind_n\}$ and ensembles $\Delta = \{e_{n+1}, \dots, e_N\}$, $\mathbf{P} = \Phi \cup \Delta$, where N represents the population size. During the population initialisation process, n single individuals Φ are first initialised by the ramped-half-and-half method [119]. Following that, the individual evaluation process gives each individual a list of case-fitnesses and standard fitness based on the current training instance by multi-case fitness evaluation [240] (details will be shown in Section 4.2.3). Then, based on the case-fitnesses of each individual, the ensemble construction and selection

process selects diverse individuals to form $N - n$ ensembles Δ (details will be shown in Section 4.2.4). In this way, we finish the population initialisation using not only individuals but also ensembles to fill the whole population.

4.2.3 Individual Evaluation

The individual evaluation process encompasses the utilisation of the multi-case fitness evaluation strategy, which assigns a list of case-fitnesses and standard fitness to each individual within the population [240]. A detailed explanation of the multi-case fitness evaluation strategy is provided in Chapter 3. For ease of reference, a concise overview of the multi-case fitness evaluation strategy is presented below.

In each generation, a new training instance consisting of n jobs is used for evaluation. To obtain the list of case-fitnesses, the instance is divided into a number of cases based on the jobs. Specifically, if we intend to divide the instance into c cases, the n jobs are split into c groups, with each group containing $g = n/c$ jobs based on their arrival time. This strategy ensures that each case contains the same group of jobs, making it fair to compare the case-fitnesses between individuals. The calculation equation is different according to the objective being optimised.

For example, when considering the mean-flowtime as the objective, the case-fitness $\text{cf}_i(x)$ for the i th case of individual ind_x is computed using Eq. (4.1). The standard fitness is determined as the mean of the case-fitness values, as depicted in Eq. (4.2).

$$\text{cf}_i(x) = \frac{1}{g} \sum_{j=g \times (i-1)+1}^{g \times i} (C_j - r_j). \quad (4.1)$$

$$sf_{\text{mean}} = \frac{1}{c} \sum_{i=1}^c \text{cf}_i. \quad (4.2)$$

where C_j and r_j represent the completion time and arrival time of the job J_j .

When targeting the max-tardiness as the objective, the case-fitness $\text{cf}_i(x)$ for the i th case of individual ind_x is calculated using Eq. (4.3). The standard fitness is computed as the maximum value among the case-fitnesses, as presented in Eq. (4.4).

$$\text{cf}_i(x) = \max_{j \in \{g \times (i-1) + 1, \dots, g \times i\}} T_j. \quad (4.3)$$

$$sf_{\max} = \max \{\text{cf}_1, \text{cf}_2, \dots, \text{cf}_c\}, \quad (4.4)$$

where T_j denotes the tardiness of job the J_j .

The case-fitnesses are used for lexicase selection, while the standard fitness is used for elitism selection and tournament selection.

4.2.4 Ensemble Construction and Selection

The *ensemble construction and selection* process selects individuals to construct ensembles. The goal of ensemble construction and selection is to choose diverse and complementary individuals for constructing each ensemble. It involves two parts: lexicase selection and similarity-checking. The pseudo-code of forming an ensemble by the ensemble construction and selection is shown in Algorithm 6. At the beginning, the ensemble \mathbf{e}_i is empty (Line 1). To achieve diverse and complementary individuals for an ensemble, we first use lexicase selection to select individuals [240] (Line 4). By lexicase selection, we expect to select a diverse set of expert individuals that are good at handling different cases. Every time a candidate individual is selected and ready to be added to the ensemble, the similarity-checking process is triggered to check whether the selected candidate individual is good at handling different cases from the individuals that are already in the ensemble.

To be specific, as the individual is selected by lexicase selection through a sequence of cases, it is expected to perform well on the cases that rank front in the case sequence. For example, we consider 5 cases and select an individual ind_a based on the case sequence $o_a = [2, 1, 5, 3, 4]$. Among

Algorithm 6: Ensemble construction and selection.

Input: Individual set: Φ ; Ensemble size: s ; Number of cases for similarity-checking: h ; Similarity threshold: ζ .

Output: An ensemble of scheduling heuristics: \mathbf{e}_i .

```

1  $\mathbf{e}_i \leftarrow \emptyset$ ;
2  $times \leftarrow 0$ ;
3 while  $size(\mathbf{e}_i) < s$  &  $times < 3 * s$  do
4    $ind_a \leftarrow LexicaseSelection(\Phi)$ ;
5    $o_a \leftarrow CaseOrder(ind_a)$ ;
   // Similarity-checking
6    $\eta_{a,b} \leftarrow 0$ ;
7   for  $ind_b \in \mathbf{e}_i$  do
8      $o_b \leftarrow CaseOrder(ind_b)$ ;
9     for  $k = 1 \rightarrow h$  do
10      if  $o_a[k] == o_b[k]$  then
11         $\eta_{a,b} \leftarrow \eta_{a,b} + \frac{c-k-1}{c} \cdot P(o_a[k], o_b[k])$ ;
12      end
13    end
14  end
15  if  $\eta \leq \zeta$  then
16     $\mathbf{e}_i \leftarrow \mathbf{e}_i \cup ind_a$ ;
17  end
18   $times \leftarrow times + 1$ ;
19 end
20 return  $\mathbf{e}_i$ ;

```

these 5 cases, this individual is expected to perform the best on the 2nd case (as $o_a[1] = 2$), followed by the 1st case (as $o_a[2] = 1$), and finally the 4th case (as $o_a[5] = 4$). Further, we believe that different cases have different importances. In this case, every time an individual is added to the ensemble, the case sequence o_a used for the lexicase selection is also recorded. Then, we check the similarity between the top h cases of the case sequence of the candidate individual and that of all the individuals in the ensemble (Line 11). Only when the individual is selected using different cases (i.e., the similarity η is smaller than a threshold ζ), it can be added

to the ensemble (Line 16). Here, the threshold ζ is set to 0, which means that we expect elements in the ensemble to be selected based on totally different top h cases of the case sequence. This approach enforces maximal diversity in ensemble creation, mitigating potential bias towards specific cases.

For example, if we consider that the first 2 cases ($h = 2$) play more important roles, then we perform the similarity-checking between individuals by comparing whether the first 2 cases are the same. Meanwhile, if we set the ensemble size to s , then when there have been s individuals added to the ensemble or $3 * s$ individual additions have been tried, the process is finished and an ensemble is obtained. Specifically, the similarity $\eta_{a,b}$ between individuals ind_a and ind_b is calculated as Eq. (4.5).

$$\eta_{a,b} = \sum_{k=1}^h \frac{c - k - 1}{c} \cdot P(o_a[k], o_b[k]) \quad (4.5)$$

where $P(o_a[k], o_b[k])$ is a decision variable, which is 1 if the k th case $o_a[k]$ of individual ind_a equal to the k th case $o_b[k]$ of individual ind_b , and 0 otherwise.

4.2.5 Ensemble Evaluation

At each generation, both single individuals and ensembles are evaluated on the same training instance(s). Since an ensemble consists of multiple individuals (i.e., scheduling heuristics), an aggregation method is required to make a decision at each decision point. In this chapter, we adopt the voting strategy [154] to make the final decision. The voting strategy selects the operation/machine that received the most votes from all the scheduling heuristics in the ensemble. Same to the individual evaluation, we use the multi-case fitness evaluation [240] to evaluate each ensemble. After ensemble evaluation, the standard fitness and a list of case-fitnesses of each ensemble are obtained.

4.2.6 Genetic Operators

During the evolutionary process, tournament selection is used to select parents, which allows both single individuals and ensembles to be chosen based on their standard fitnesses. Crossover and mutation operators are commonly used to generate offspring in GP. We develop novel crossover and mutation operators, which can be applied to both single individuals and ensembles to generate offspring. In addition, in order to explore a wider range of ensemble combinations, the proposed ensemble construction and selection in Section 4.2.4 enables the generation of ensemble offspring by selecting individuals from the current generation to form ensembles. The details about the newly proposed crossover and mutation operators are shown as follows.

Crossover

Crossover requires two parents. In our proposed EGP^e method, the selected two parents have three possible combinations, i.e., individual and individual, individual and ensemble, or ensemble and ensemble:

- If the two parents are both individuals, the conventional tree swapping crossover [265] is adopted to generate two offspring, and the two offspring are individuals.
- If one parent is an individual and the other is an ensemble, then one offspring is generated by randomly selecting an individual from the ensemble to do tree swapping crossover between this individual and the other individual parent. The generated offspring is an individual. The other offspring is generated by randomly replacing an individual of the ensemble with the individual parent. This offspring is an ensemble. The detailed process is shown in Figure 4.3.
- If the two parents are both ensembles, then two offspring are generated by randomly selecting an individual from each ensemble and

swapping them. Both offspring are ensembles. The detailed process is shown in Figure 4.4.

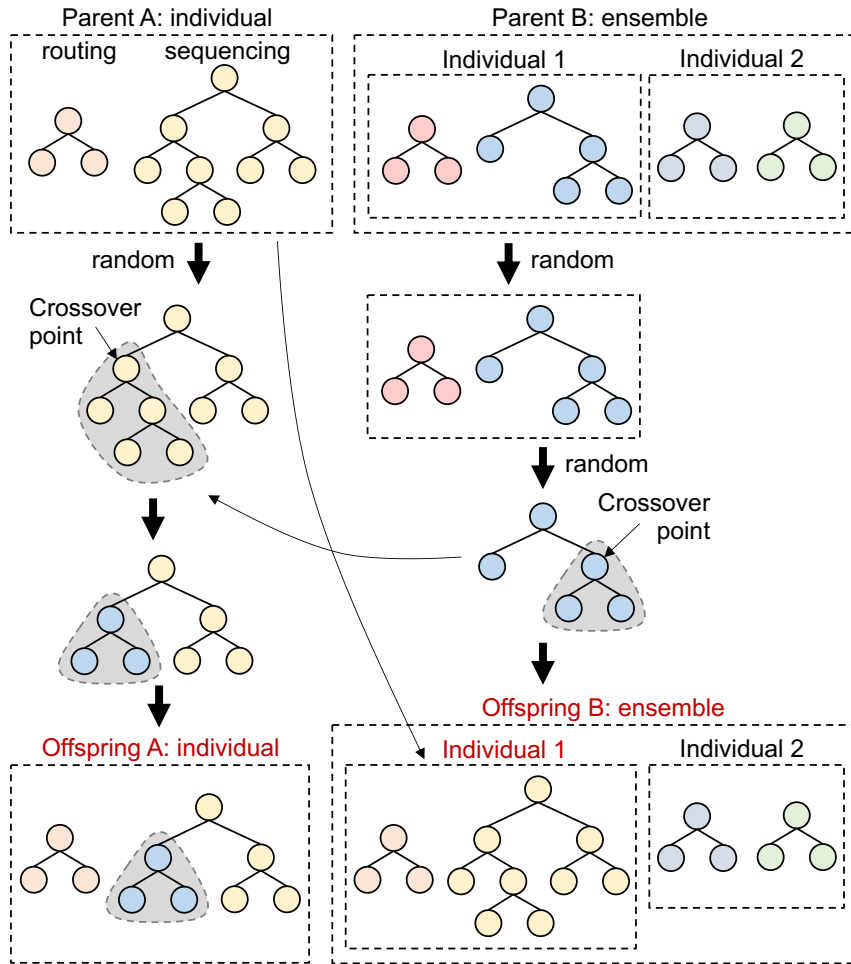


Figure 4.3: The crossover process between an individual and an ensemble.

Mutation

Mutation generates offspring from a single parent. There are two situations depending on the parent, i.e., individual or ensemble:

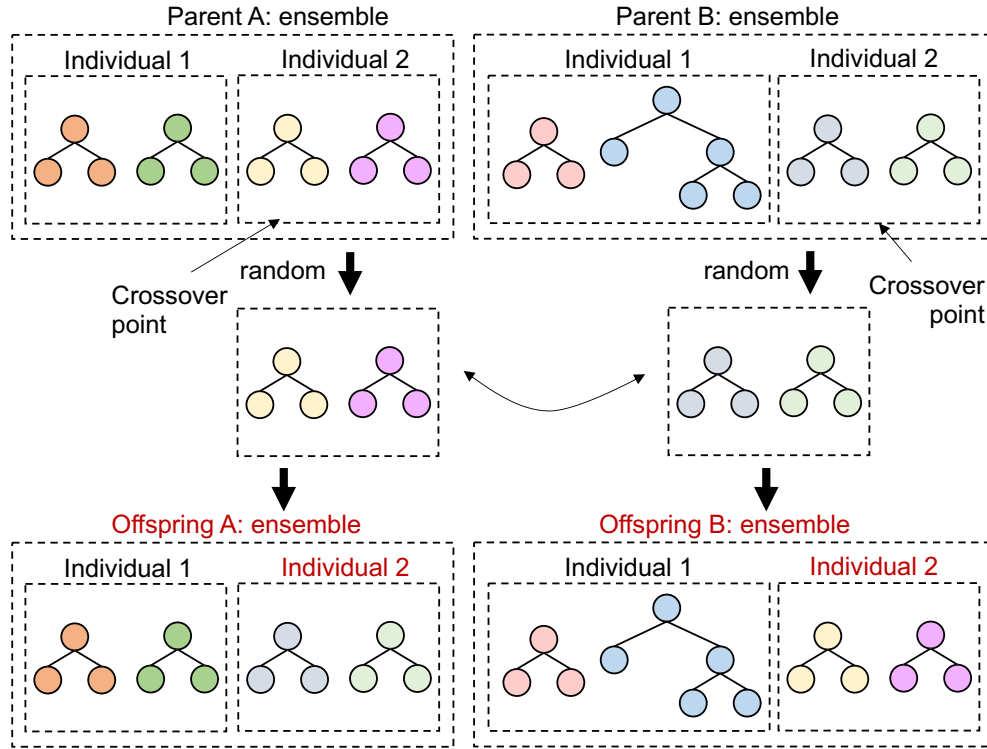


Figure 4.4: The crossover process between two ensembles.

- If the parent is an individual, then the standard subtree mutation [265] is adopted to generate offspring, and the offspring is also an individual.
- If the parent is an ensemble, then we randomly select an individual from the ensemble and apply the standard subtree mutation to generate an offspring. The generated offspring is an individual. The detailed process is illustrated in Figure 4.5.

Note that for mutation, the generated offspring is a single individual regardless of whether the parent is an individual or an ensemble. The reason for allowing only individuals rather than ensembles as offspring is that in addition to producing ensemble offspring that can be inherited to the next generation, we also want to explore more ensemble structures

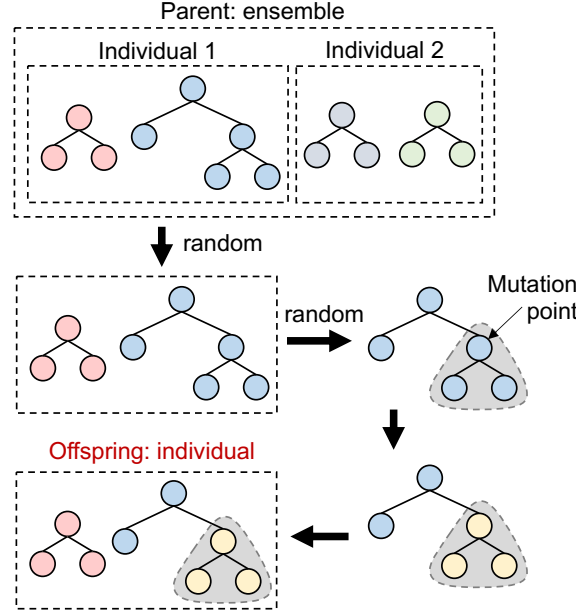


Figure 4.5: The mutation process for an ensemble parent.

by randomly combining the generated offspring to form different ensembles based on the proposed ensemble construction and selection strategy. Therefore, in order to leave some place for new ensembles while maintaining the ratio of single individuals and ensembles in the population at the same time, only individuals are allowed as offspring for mutation. In addition, to further maintain the same ratio of single individuals and ensembles in the population, the proposed method limits the number of ensembles to a fixed number $N - n$. That is, when the generated ensembles reach $N - n$, the proposed method ignores the generated ensembles and only keeps the generated individuals as offspring.

4.2.7 Computational Complexity

To help understand the efficiency of the proposed algorithm, especially when dealing with the large-scale and complex DFJSS problem. We analyse the computational complexity. The computational complexity of the

classical GP algorithm mainly depends on the following factors: population size (N), number of generations (G), and the complexity of fitness evaluation (F). In each generation, the GP algorithm performs various operations like selection, crossover, mutation, and fitness evaluation. The computational complexity of these operations is typically linear with respect to the population size and the complexity of fitness evaluation. When considering the classical GP algorithm applied to the large-scale DFJSS problem. The computational complexity of fitness evaluation is equal to the computational complexity of simulating a DFJSS instance, which mainly depends on the number of decision points D required to process. Thus, the computational complexity of fitness evaluation (F) can be approximated as (D). In each generation, the computational complexity of other steps such as crossover, mutation, and parent selection are much smaller than that of the fitness evaluation, and thus can be ignored. Overall, the total computational complexity of classical GP for solving the large-scale DFJSS problem can be approximated as $O(N * G * D)$.

In the proposed EGP^e , the newly proposed ensemble construction and selection strategy may add some extra complexity to the classical GP. However, the impact on training time is relatively small. The computational complexity of the ensemble construction and selection process, as shown in Algorithm 6, mainly depends on the ensemble size (s) and lexicase selection (L). The worst-case computational complexity for a single lexicase selection is $p * c$, where p denotes the candidate pool size and c is the number of cases. Specifically, the computational complexity for constructing an ensemble can be approximated as $s * p * c$, which is significantly lower than the computational complexity of fitness evaluation of the DFJSS problem ($s * p * c \ll D$), which involves thousands of decision points D . Moreover, if jobs arrive frequently, the waiting queue for each machine will expand, leading to more decision points and significantly increasing the computational complexity of evaluating a DFJSS instance. Hence, the proposed EGP^e would not significantly increase the overall

training computational compared to the classical GP. The total computational complexity of using EGP^e for solving the large-scale DFJSS problem is still $O(N * G * D)$, which is the same as the classical GP.

4.3 Experimental Design

The dataset employed for simulating various job shop scheduling scenarios, along with the terminals and functions utilised for constructing individuals in this chapter, remains consistent with Chapter 3. The configuration of the improved and compared GP methods and experimental design of this chapter are outlined as follows.

4.3.1 Parameter Setting

The parameter configurations for all GP methods can be found in Table 4.1. Specifically, an ensemble size of 5 is set for EGP^e , as previous research has demonstrated that this ensemble size is able to provide promising performance, and increasing the ensemble size does not yield significant improvements in results [210, 213].

4.3.2 Comparison Design

In this chapter, we consider six scenarios by considering two different utilisation levels (0.85 and 0.95) and three different objectives ($Fmax$, $Tmax$, and $WTmean$). To validate the effectiveness of the presented EGP^e method, we compare it against the following algorithms:

1. **GP**: the standard multi-tree GP proposed for the DFJSS problem that outputs an individual as its final result [265].
2. **BagGP**: the Bagging GP that evolves an element each time using a different subset of training instances. It finally outputs the group of all the evolved elements as an ensemble [209].

Table 4.1: The configuration of parameters for GP methods.

Parameter	Value
Initialisation method	Ramped-half-and-half
Initial maximum/minimum tree depth	6 / 2
Population size	1000
Maximal number of evaluations	50000
Maximal tree depth	8
Elitism individual/ensemble	8 / 2
Crossover/mutation rate	0.8 / 0.2
Non-terminal/terminal selection rate	0.9 / 0.10
Parent selection	Tournament selection with size 7
Ensemble construction and selection	Lexicase selection
Ensemble from offspring probability	0.3

3. **CCGP**: the cooperative coevolution GP with 5 subpopulations, each has 200 individuals [176]. The final output ensemble consists of the best individuals in the 5 subpopulations.
4. **DivNichGP**: the GP with a niching technique and a greedy ensemble selection strategy [222]. It uses a niching technique to increase population diversity and evolve a diverse set of effective individuals. Then, the ensemble is formed by selecting the individuals in the final population in a greedy way (from the best to the worst). To be noticed, DivNichGP outputs the best individual during the evolution process and then outputs the ensemble every 1000 evaluations when doing the greedy selection.
5. **M3GP**: the multidimensional multiclass GP with multidimensional populations [18]. It starts with one scheduling heuristic per ensemble and uses three newly designed operators for removing/adding/swapping scheduling heuristic(s) from/to/between ensemble(s). It finally outputs the best-evolved ensemble. To be noticed, we do not restrict the evaluation times to 50000, and we

use crossover and mutation rates of 0.5, following the settings of the original paper [18].

6. **eGP**: the ensemble GP [187]. It has 2 subpopulations, one for evolving scheduling heuristics and the other for evolving ensembles. The evolution of the subpopulation with individuals is the same as classical GP. It proceeds the evolution of ensembles by removing/adding/swapping scheduling heuristic(s) from/to/between ensemble(s). It finally outputs the best-evolved ensemble. We adopt the same parameter settings as M3GP, in accordance with the configurations presented in the original paper [187].

The reasons to select these algorithms as the comparison methods are because: 1) GP is a typical method to the DFJSS problem, and is widely used as a benchmark in this field; 2) BagGP is a variation that applies a classical ensemble learning technique (Bagging) to GP; 3) CCGP, DivNichGP, M3GP, and eGP are effective GP-based methods for evolving ensembles of heuristics.

4.4 Results and Discussions

4.4.1 Sensitivity Analysis

To examine the impact of the number of ensembles on the proposed EGP^e method, a sensitivity analysis was conducted. Figure 4.6 presents violin plots illustrating the test performance of 30 runs of the proposed EGP^e method with varying numbers of ensembles (20, 40, 60, 80, and 100) in the population across six scenarios. To be noticed, the number of single individuals in the population changes as the number of ensembles changes. For example, when we consider 20 ensembles, the number of single individuals in the population will be $1000 - 20 \times 5 = 900$. The black curves in the figure represent the trend of the mean test performance as the num-

ber of ensembles increases. It is evident that the effect of the number of ensembles on the test performance varies across different scenarios.

Moreover, Table 4.2 provides the mean and standard deviation of the test performance for EGP^e using different numbers of ensembles in the population across 30 independent runs in the six scenarios. To rank the EGP^e method with varying numbers of ensembles, we conducted the Friedman test [287]. The results indicate that the EGP^e method with 40 ensembles achieved the highest rank, followed by 60, 20, 80, and 100 ensembles, respectively. Hence, we observe that the EGP^e method with 40 ensembles delivers the best performance among the tested parameters. Subsequent sections will focus on further analyses using the EGP^e method with 40 ensembles. To simplify the description, in the subsequent sections we use EGP^e to denote EGP^e with 40 ensembles.

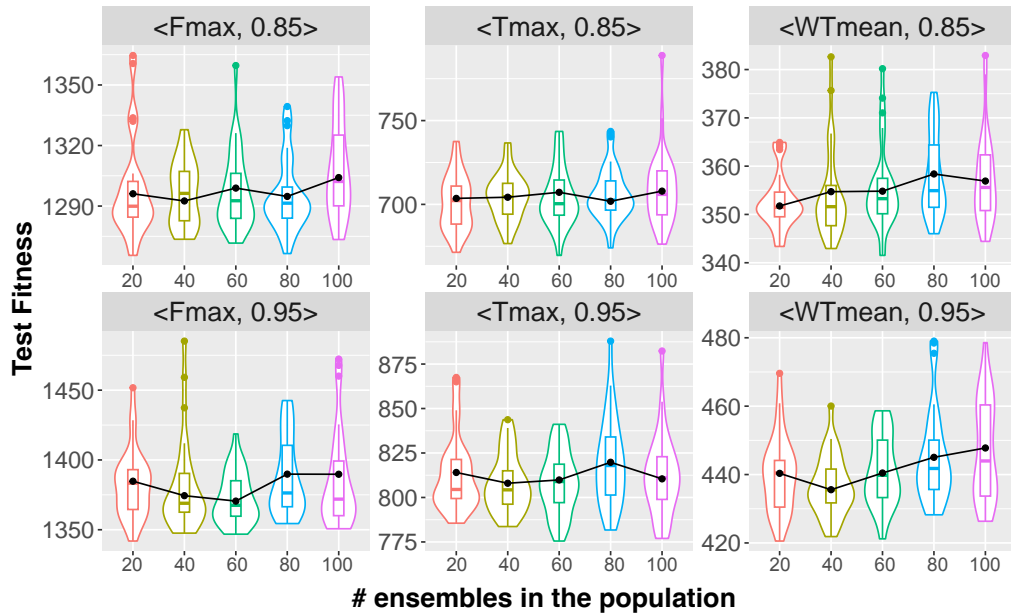


Figure 4.6: The violin plots of the test performance of 30 runs of the proposed EGP^e with different ratios of single individuals and ensembles on 6 scenarios.

Table 4.2: The mean and standard deviation test performance of the EGP^e method with different numbers of ensembles in the population through 30 independent runs across 6 scenarios.

S^*	EGP^e -20	EGP^e -40	EGP^e -60	EGP^e -80	EGP^e -100
1*	1297.57(24.33)	1296.30(15.68)	1296.95(18.93)	1294.92(17.83)	1307.81(23.21)
2*	1382.50(24.17)	1380.17(32.67)	1372.36(18.27)	1387.91(27.49)	1385.42(34.20)
3*	700.98(16.23)	704.40(13.83)	704.29(17.65)	705.48(15.23)	709.33(21.96)
4*	812.95(20.88)	807.93(16.10)	808.15(17.46)	819.59(24.45)	812.00(22.79)
5*	352.60(5.69)	354.06(9.25)	355.33(8.71)	358.09(8.45)	357.10(8.93)
6*	440.00(11.06)	436.66(8.59)	441.02(10.15)	445.02(13.91)	447.20(14.51)
Rank	2.5	1.83	2.33	4	4.33

* S : Scenarios, 1: $\langle F_{max}, 0.85 \rangle$, 2: $\langle F_{max}, 0.95 \rangle$, 3: $\langle T_{max}, 0.85 \rangle$, 4: $\langle T_{max}, 0.95 \rangle$, 5: $\langle WT_{mean}, 0.85 \rangle$, 6: $\langle WT_{mean}, 0.95 \rangle$.

4.4.2 Test Performance

Table 4.3 gives the mean and standard deviation of the test performance derived from 30 independent runs of the EGP^e method and compared methods across six DFJSS scenarios. We employ the Wilcoxon rank sum test [54] to perform comparisons, where EGP^e is compared with GP, BagGP, CCGP, DivNichGP, M3GP, and eGP, respectively. The results are considered significant when the obtained p -value is less than 0.05 and a smaller (better) mean test performance is given. The symbols “ $\uparrow/\downarrow/=$ ” next to the results of EGP^e in the table indicate whether the results are significantly better than, worse than, or comparable to the results of each algorithm located to the left of EGP^e . In the case of the $\langle F_{max}, 0.85 \rangle$ scenario (first row of the table), the $(=)(\uparrow)(\uparrow)(\uparrow)(\uparrow)(\uparrow)$ for EGP^e indicates that EGP^e shows no significantly different performance from that of GP, performs significantly better than BagGP, CCGP, DivNichGP, M3GP, and eGP. In addition, the Friedman test [287] is conducted to rank these methods.

From Table 4.3, it is observed that EGP^e performs significantly better than DivNichGP on two scenarios and is comparable to DivNichGP on the remaining four scenarios. Compared to standard GP, EGP^e out-

Table 4.3: The mean and standard deviation performance of the EGP^e and compared methods on unseen test instances through 30 independent runs across 6 scenarios.

Scenario	GP	BagGP	CCGP	DivNichGP
<Fmax, 0.85>	1304.36(23.96)	1338.32(20.62)	1354.59(36.46)	1305.82(15.66)
<Fmax, 0.95>	1388.71(27.11)	1446.07(31.94)	1472.51(87.37)	1389.39(27.44)
<Tmax, 0.85>	711.58(14.85)	742.77(13.18)	768.02(40.37)	705.16(14.90)
<Tmax, 0.95>	816.22(18.08)	874.25(12.31)	899.91(71.35)	813.72(20.42)
<WTmean, 0.85>	354.89(9.53)	370.13(6.23)	361.58(14.27)	352.62(7.33)
<WTmean, 0.95>	440.69(10.79)	457.32(5.26)	446.36(10.04)	439.35(9.10)
Improvement	0.73%	4.88%	5.56%	0.40%
Average rank	2.67	5.33	5.67	2.17

Scenario	M3GP	eGP	EGP ^e
<Fmax, 0.85>	1558.49(72.74)	1322.48(25.00)	1296.30(15.68)(=)(↑)(↑)(↑)(↑)(↑)
<Fmax, 0.95>	1768.12(298.25)	1418.06(52.46)	1380.17(32.67)(↑)(↑)(↑)(↑)(↑)(↑)
<Tmax, 0.85>	918.16(54.20)	722.19(24.10)	704.40(13.83)(↑)(↑)(↑)(=)(↑)(↑)
<Tmax, 0.95>	1082.08(99.83)	833.77(23.87)	807.93(16.10)(↑)(↑)(↑)(=)(↑)(↑)
<WTmean, 0.85>	430.99(13.87)	358.15(8.29)	354.06(9.25)(=)(↑)(↑)(=)(↑)(↑)
<WTmean, 0.95>	531.80(25.51)	443.74(11.48)	436.66(8.59)(=)(↑)(↑)(=)(↑)(↑)
Improvement	20.52%	2.16%	-
Average rank	7.0	4.0	1.17

performs it on three out of six scenarios. For the remaining three scenarios, EGP^e performs similarly to GP but with better mean test performance and smaller standard deviation. Moreover, EGP^e significantly outperforms BagGP, CCGP, M3GP, and eGP on all six scenarios. In addition, the outcomes of the Friedman test presented in Table 4.3 indicate that EGP^e ranked first, followed by compared methods with the order DivNichGP, GP, eGP, BagGP, CCGP, and M3GP. More precisely, the average improved percentage ρ of the proposed EGP^e over each comparison method A across the 6 scenarios are calculated based on the Eq. (4.6) and shown at the bottom of Table 4.3. The reason why BagGP, CCGP, M3GP, and eGP perform worse than the proposed EGP^e and GP is that these methods do not con-

sider the problem-specific characteristics of DFJSS. They neglect the commonly used seed rotation strategy in DFJSS, which plays a crucial role in its optimisation process.

$$\rho(EGP^e|A) = \frac{1}{6} \sum_{i=1}^6 \frac{(Obj(A) - Obj(EGP^e))}{Obj(A)} \quad (4.6)$$

To further compare the proposed EGP^e with standard GP and DivNichGP which are the second and third places, Figure 4.7 shows their convergence curves of the mean test performance of 30 runs from the number of evaluations 5000 to the number of evaluations 50000 on 6 scenarios. As we can see, compared to standard GP, EGP^e converges faster and ultimately converge to better results on scenarios $\langle T_{max}, 0.85 \rangle$, $\langle F_{max}, 0.95 \rangle$, and $\langle T_{max}, 0.95 \rangle$. For the remaining scenarios, the curves of standard GP and EGP^e are quite close to each other but EGP^e finally gives smaller (better) test performance when arriving at 50000 evaluations. Compared to DivNichGP, except for the scenario $\langle WT_{mean}, 0.85 \rangle$, EGP^e converges faster and gives better test performance on most of the generations on the remaining scenarios. Overall, the results confirm that the proposed EGP^e method is more effective than standard GP, BagGP, CCGP, DivNichGP, M3GP, and eGP.

4.4.3 Effectiveness of Ensemble Construction

To study the effect of the proposed ensemble construction and selection on the proposed method, we compare the proposed EGP^e method to a variation of EGP^e that uses the standard tournament selection to select elements to form ensembles (not using the proposed ensemble construction and selection), which is named EGP^t . Table 4.4 provides the mean and standard deviation test performance resulting from 30 independent runs of both EGP^t and EGP^e across 6 scenarios. Notably, EGP^e outperforms EGP^t on 3 scenarios and demonstrates no statistical difference in comparison to EGP^t within the remaining 3 scenarios. Moreover, on the

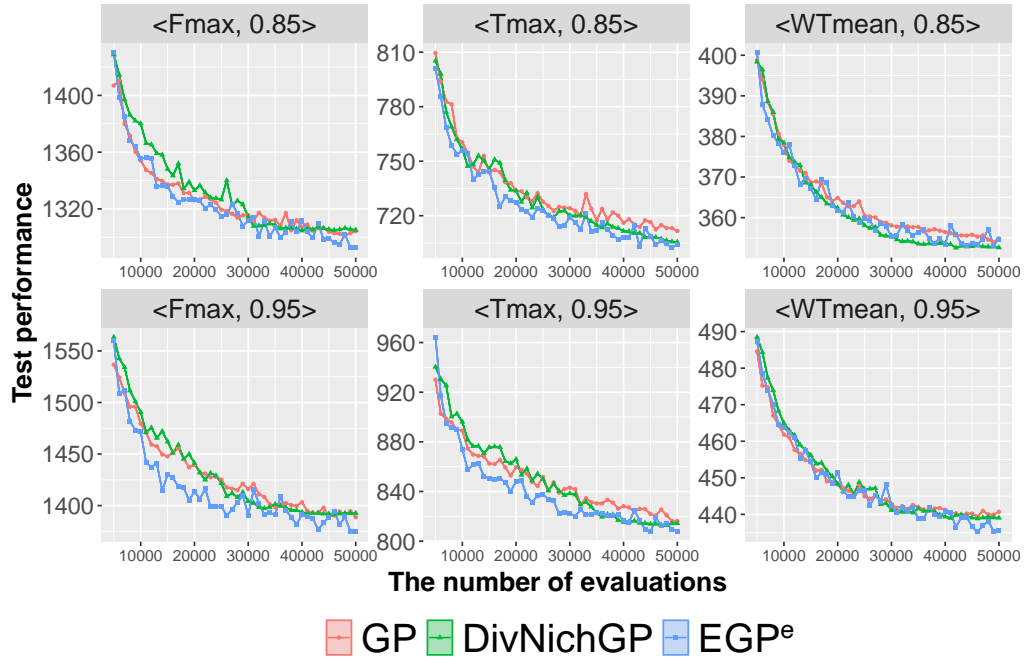


Figure 4.7: The convergence curves of the mean test performance of 30 runs of standard GP, DivNichGP, and EGP^e on 6 scenarios.

3 scenarios with similar performance, the proposed EGP^e method gives numerically better performance (smaller mean test objective value) than EGP^t . Through the comparison of these two methods, the effectiveness of the proposed ensemble construction and selection strategy is verified.

Table 4.4: The mean and standard deviation of test performance from 30 independent runs of EGP^t and EGP^e across 6 scenarios.

Scenario	EGP^t	EGP^e
<Fmax, 0.85>	1306.09(23.11)	1296.30(15.68)(=)
<Fmax, 0.95>	1421.66(184.43)	1380.17(32.67)(↑)
<Tmax, 0.85>	710.56(16.87)	704.40(13.83)(=)
<Tmax, 0.95>	820.41(21.94)	807.93(16.10)(↑)
<WTmean, 0.85>	354.21(9.22)	354.06(9.25)(=)
<WTmean, 0.95>	440.41(8.94)	436.66(8.59)(↑)

4.4.4 Effectiveness of Genetic Operators

To test the effectiveness of the presented genetic operators that consider single individuals and ensembles together, we conducted experiments comparing EGP^e to a variation of EGP^e with classical crossover and mutation operators, which we named EGP^c . The EGP^c method only allows individuals to be parents and produce individual offspring, while the proposed ensemble construction and selection strategy is still used to populate the ensemble portion of the population. Table 4.5 presents the mean and standard deviation test performance by 30 runs of EGP^c and EGP^e on 6 scenarios. The results show that EGP^e outperforms EGP^c on 2 scenarios and performs comparably on the other 4 scenarios. These findings confirm the effectiveness of the presented genetic operators (new crossover and mutation) in generating high-quality offspring by leveraging the advantages of both single individuals and ensembles.

Table 4.5: The mean and standard deviation of test performance from 30 independent runs of EGP^c and EGP^e across 6 scenarios.

Scenario	EGP^c	EGP^e
<Fmax, 0.85>	1302.21(25.29)	1296.30(15.68)(=)
<Fmax, 0.95>	1388.67(31.58)	1380.17(32.67)(=)
<Tmax, 0.85>	707.28(13.74)	704.40(13.83)(=)
<Tmax, 0.95>	820.45(24.65)	807.93(16.10)(↑)
<WTmean, 0.85>	357.16(7.86)	354.06(9.25)(↑)
<WTmean, 0.95>	440.69(12.11)	436.66(8.59)(=)

4.5 Further Analyses

4.5.1 Performance of Ensemble and Elements

We typically expect an ensemble to perform better than each individual element within it. In this case, to analyse the performance relationship

between the ensemble and its individual elements, we create the scatter plots (x, y) with the ensemble performance as the x and the element performance as the y . The visualisation is based on the 30 runs from the number of evaluations 40000 to 50000, where EGP^e produces the ensemble as the best solution for every 1000 evaluations. The scatter plots of the training performance of the ensemble versus the training performance of the element from the number of evaluations 40000 to 50000 of 30 runs by EGP^e on 6 scenarios are shown in Figure 4.8. To be noticed, the points in red colour represent that the ensemble outperforms the corresponding individual element and the line denotes the reference line of $y = x$ which makes it easy to see the performance relationship between the ensemble and its individual elements. It can be seen from Figure 4.8, the learned ensembles always outperform their individual element on the training instances.

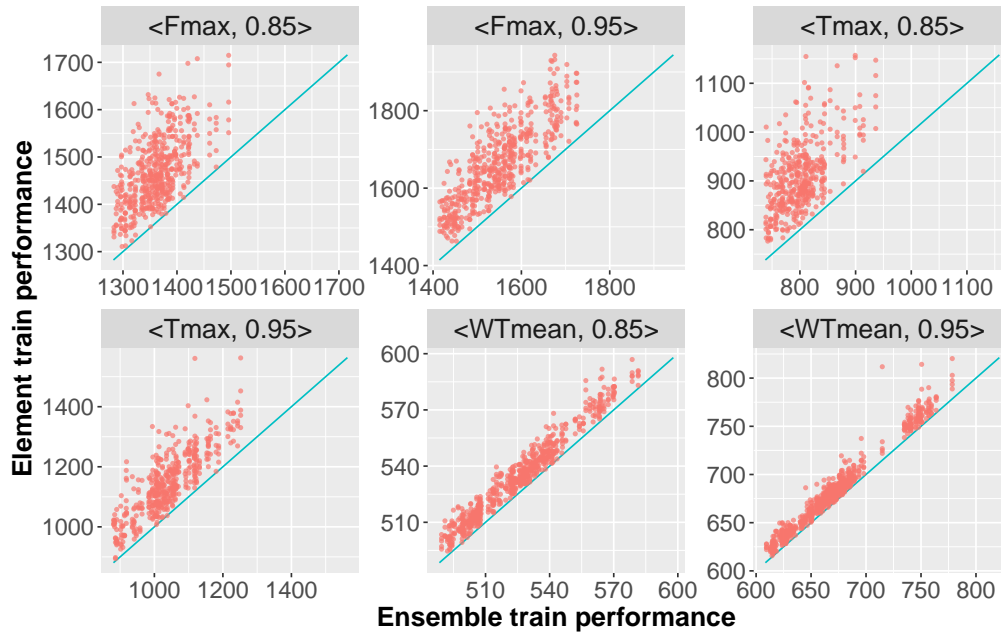


Figure 4.8: The scatter plots of the train performance of the ensemble versus that of its each element from the number of evaluations 40000 to 50000 of 30 runs by EGP^e on 6 scenarios.

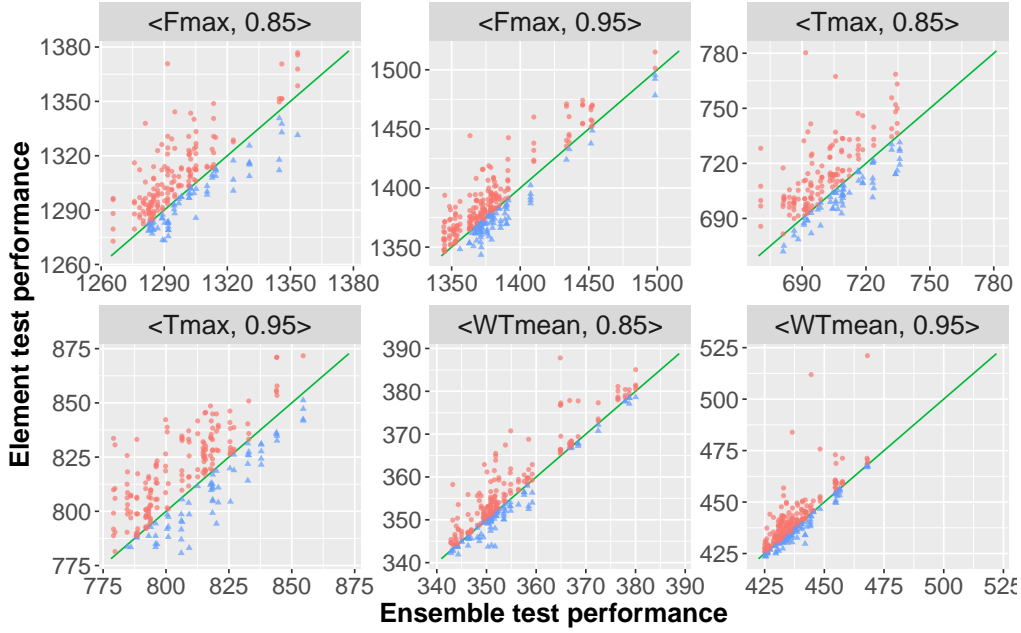


Figure 4.9: The scatter plots of the test performance of the ensemble versus the test performance of its each element from the number of evaluations 40000 to 50000 of 30 runs by EGP^e on 6 scenarios.

However, we observe different phenomena on unseen instances. Figure 4.9 shows the scatter plots of the test performance of the ensemble versus the test performance of the element from the number of evaluations 40000 to 50000 of 30 runs by EGP^e on 6 scenarios. Similarly, the points in red colour represent that the ensemble outperforms the corresponding individual element, while the points in blue colour indicate that the ensemble performs worse than the corresponding individual element. As we can see, it is possible for an individual element to outperform the ensemble it belongs to on the unseen test set. Nevertheless, the visualisation indicates that in most cases, the ensemble still outperforms the individual element by a significant margin.

This finding emphasises the importance of carefully selecting elements to ensure that the formed ensemble can achieve better performance. This

also suggests that a combination of single individuals and ensembles is necessary to achieve good performance. Moreover, it suggests that the ensemble construction and selection process may not fully capture the underlying characteristics of the problem for the unseen data, which could be explored in future research to improve the ensemble construction and selection strategy.

4.5.2 Ensemble Win Percentage

The proposed method incorporates both single individuals and ensembles in its population, it is interesting to track which type of solution, i.e., individual or ensemble, performs better at each generation, as well as the percentage of times each type of solution achieves the best performance across 30 runs.

Figure 4.10 illustrates the percentage of times an ensemble achieves the best performance every 1000 evaluations of EGP^c and EGP^e across six scenarios of 30 runs. For different scenarios, we observe different phenomena. For the scenarios with max-objective, at the beginning, the percentage of ensembles that achieve the best solution is low, but as evolution progresses, the percentage of ensembles that achieve the best solution increases, reaching a peak of approximately 50% around at sixth generation. Subsequently, the percentage of ensembles achieving the best solution decreases but remains stable at around 10% towards the end of evolution. For the scenarios with mean-objective, a similar trend in the curve was observed, but the corresponding values were relatively large. To be specific, initially, the proportion of ensembles that reach the best solution is low. However, as evolution continues, this proportion gradually increases, peaking at approximately 80% by the sixth generation. Subsequently, the percentage of ensembles attaining the best solution declines but stabilises at around 20% towards the end of the evolutionary process. This is a high proportion considering that the proportion of ensembles in the entire pop-

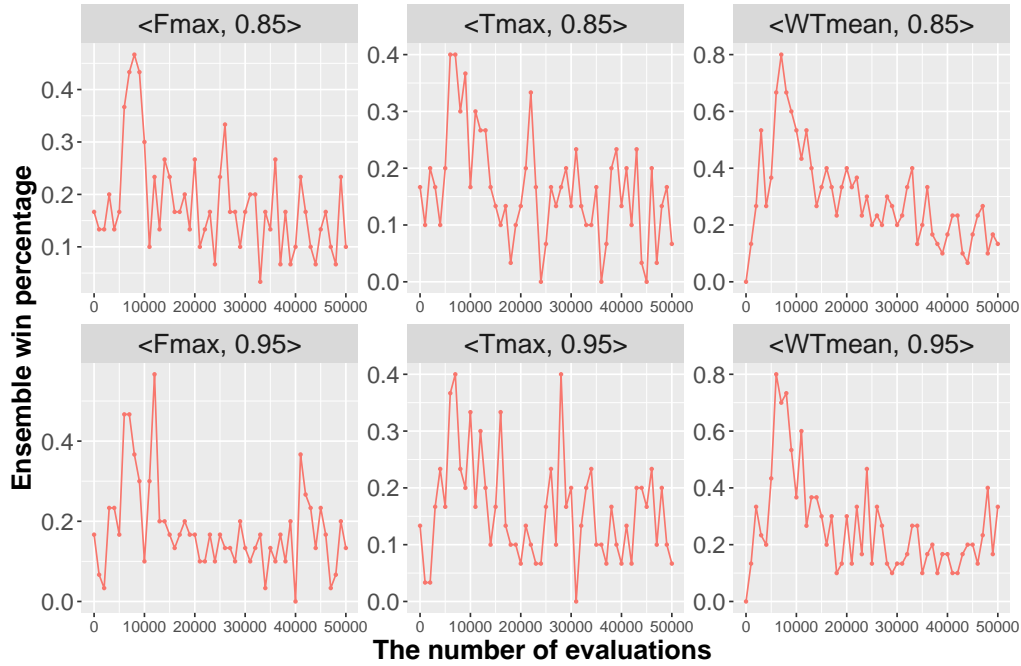


Figure 4.10: The convergence curves of the percentage of the ensemble is output as the best solution every 1000 evaluations on 6 scenarios of 30 runs by the proposed EGP^e.

ulation is only 4.8%. By tracking the performance of each solution type and the percentage of times each type achieves the best performance, we suggest that a dynamic adjustment strategy to change the proportion of ensembles and individuals in the population can help to optimise the population and improve overall solution quality, which can be explored in the future. In addition to this, the above results show that the ensemble has a more pronounced role on the scenarios with mean-objective. In this case, we suggest to further analyse the difference between different objectives and design specific strategies for different objectives.

Furthermore, to provide a more intuitive understanding of the results, we present the number of runs in which ensembles are significantly better, worse, or comparable to individuals in the population out of 30 runs every

1000 evaluations on six scenarios of EGP^e, which is shown as Figure 4.11. It can be seen that the distinction is not only scenario-dependent but also changes with the evolutionary process. To be specific, in scenarios with a max-objective, ensembles outperform individuals in most cases during the early stages of evolution. As the evolution progresses, the frequency of ensembles outperforming individuals decreases, while the frequency of ensembles behaving similarly to individuals increases. In the late stages of evolution, ensembles perform significantly better than individuals in a few cases and have no statistical difference from individuals in most cases. However, in the scenario with a mean-objective (i.e., the bottom graph in Figure 4.11), ensembles perform better than individuals in most cases at the early stages of evolution. In the middle stages of evolution, ensembles mostly perform significantly worse than individuals, while in the late stages of evolution, ensembles exhibit no significant difference from individuals in most cases.

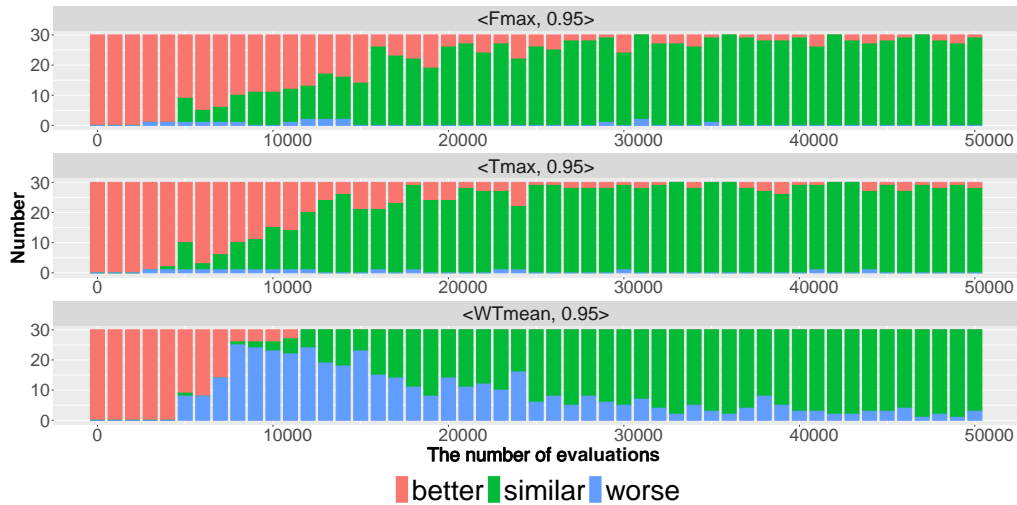


Figure 4.11: The number of runs that ensembles are significantly better and worse than, or comparable to individuals out of 30 runs every 1000 evaluations on 3 scenarios of EGP^e.

More detailed, Figure 4.12 illustrates the fitness distribution of single

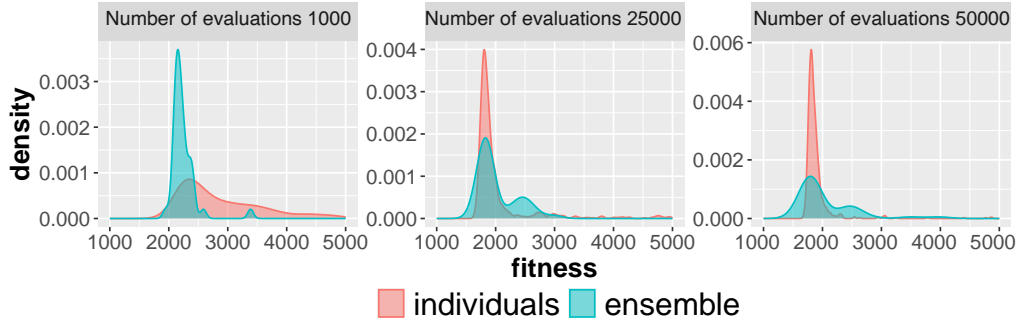


Figure 4.12: The fitness distribution of ensembles and individuals of a single run at the beginning (number of evaluations 1000), middle (number of evaluations 25000), and late (number of evaluations 50000) stages of evolution on the scenario $\langle F_{\max}, 0.95 \rangle$ of EGP^e .

individuals and ensembles from a single run at the early (the number of evaluations 1000), middle (the number of evaluations 25000), and late (the number of evaluations 50000) stages of evolution of EGP^e on the $\langle F_{\max}, 0.95 \rangle$ scenario. The range of fitness values displayed in the figure is limited to 1000 to 5000, as most single individuals and ensembles fall within this range. As observed in Figure 4.12, at the beginning stage of evolution, the shape of the distribution of fitness of ensembles is higher and leaner than individuals and performs better overall, while as the evolution process goes on, the shape of the distribution of fitness of ensembles becomes shorter and wider than that of individuals. At the same time, the overall density of ensembles' fitness is relatively uniformly distributed, rather than being very concentrated within a certain range, as is the case with individuals. Given the above analyses, we obtain the following observations:

1. Firstly, even in the early stages of evolution, when individuals tend to perform poorly, ensembles can still achieve better performance by combining them;
2. Secondly, the better performance achieved by ensembles can help to

provide a lower bound for optimisation and thus is able to improve the overall population performance of both single individuals and ensembles;

3. Thirdly, the wider and shorter density distribution of ensemble fitness as evolution proceeds indicates that ensembles can provide more stable performance than individuals;
4. Finally, when focusing on the proportion of individuals/ensembles that perform exceptionally well (i.e., fitness within the range of 1000 to 2000), ensembles still have a higher proportion than individuals. This observation further confirms the effectiveness of ensembles as an effective technique for improving solution quality.

4.5.3 Elements Contribution to Ensemble

To avoid the situation that one individual dominates all the others or the situation that other individuals make no contributions to the ensemble, it is interesting to study the ensemble contribution of each individual in an ensemble. Here, the *ensemble contribution* δ_i represents the percentage of the same decisions made by an individual element ind_i compared to the decisions made by the ensemble e_j across all decision points [175], which can be calculated by Eq. (4.7).

$$\delta_i = \frac{\sum_{d=1}^{D_j} Z_{i,j,d}}{D_j} \quad (4.7)$$

where D_j represents the total number of decisions when e_j works for the training instance(s). $Z_{i,j,d}$ is a decision variable, equal to 1 when ind_i gives the same decision with e_j on the d th decision point.

If an individual has an ensemble contribution of 0 (no same decision), it means that the individual does not make any contribution to the ensemble. Conversely, if an individual has an ensemble contribution of 1, it implies that this individual is dominating the ensemble. We expect each individual to give a high ensemble contribution, also it is expected that each

individual gives a relatively consistent (similar) ensemble contribution on training instances (training ensemble contribution) and on unseen test instances (test ensemble contribution). Figure 4.13 gives the point plots of training ensemble contribution versus test ensemble contribution of each individual in an ensemble when the ensemble is output as the best solution at each generation by the proposed EGP^e on 6 scenarios of 30 runs. The points in red colour represent that the test ensemble contribution is higher than the training ensemble contribution, while the points in blue colour indicate that the test ensemble contribution is lower than the training ensemble contribution. Also, the line denotes the reference line of $y = x$, which makes it easy to see the relationship between the training ensemble contribution and the test ensemble contribution of each individual. It can be seen that, EGP^e can evolve ensembles wherein each element can support a high training and test ensemble contribution (higher than 0.89). This gives evidence that every element in ensembles contribute to the performance of the ensemble. Also, about half of the situations where the training ensemble contribution is higher than the test ensemble contribution, and half of the situations where it is the opposite. In general, the figure shows that each point lies near the line $y = x$, which means each individual can provide a test ensemble contribution that is generally consistent with its training ensemble contribution. This finding suggests that we can trust the training results and use the trained ensembles on unseen instances.

4.5.4 Diversity

Diversity plays an important role not only in the population but also in the ensemble. In this chapter, the proposed method holds a population containing both single individuals and ensembles, and most of the ensembles are formed by individuals in the population using the developed ensemble construction and selection strategy. This section explores the di-

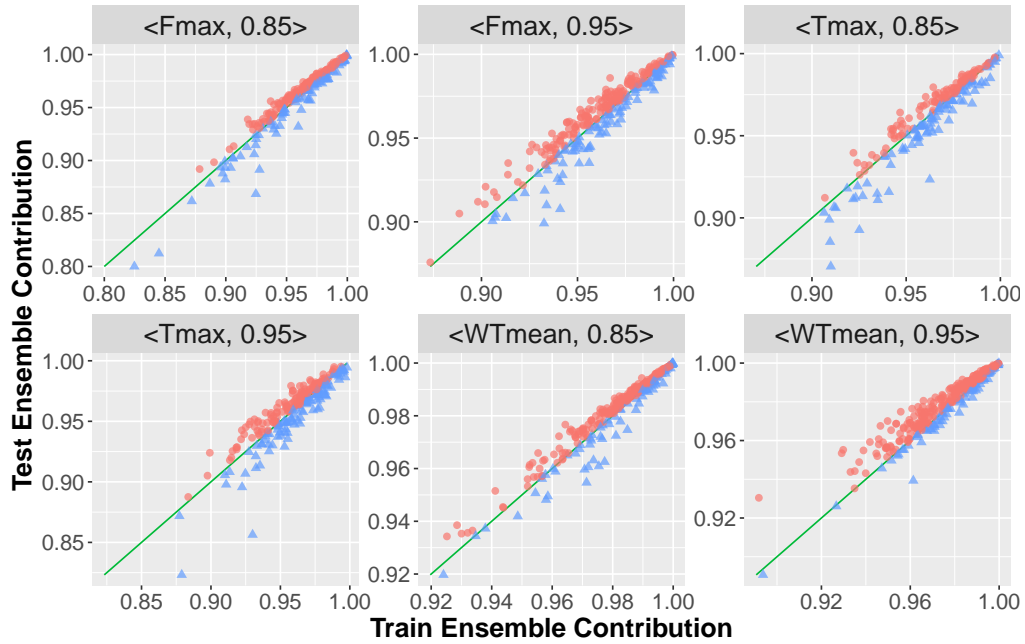


Figure 4.13: The scatter plots of training ensemble contribution versus test ensemble contribution of each individual in the learned ensembles from the number of evaluations 40000 to 50000 by the proposed EGP^e on 6 scenarios of 30 runs.

versity among individuals within the population. Within the domain of DFJSS, *phenotypic diversity* is more meaningful than *genotypic diversity*, as scheduling heuristics that have different structures (genotypes) can have the same behaviour (phenotype). Here, the phenotypic diversity is quantified through the calculation of the distinct sets of decisions made by individuals within the population across 20 sequencing and 20 routing decision points [238].

Figure 4.14 shows the convergence curves of the phenotypic diversity on each generation by the proposed EGP^e and EGP^t on 6 scenarios of 30 runs. It can be observed that the proposed EGP^e gives a higher phenotypic diversity on almost all the generations on 5 of the scenarios, except for the scenario $\langle WTmean, 0.95 \rangle$. Analysing these results with the test perfor-

mance of EGP^e and EGP^t from Section 4.4.3, it shows that higher phenotypic diversity is able to contribute to obtaining good performance on the scenarios with max-objective, while not contribute to obtaining good performance on the scenario with mean-objective.

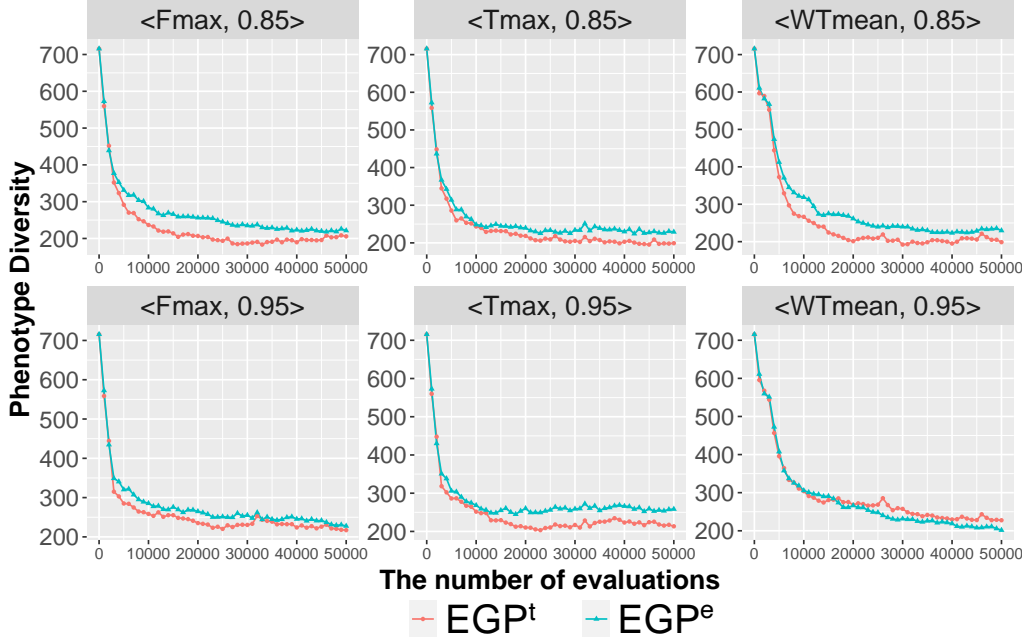


Figure 4.14: The convergence curves of the phenotypic diversity of the individuals in the population by the proposed EGP^e and EGP^t on 6 scenarios of 30 runs.

This observation implies that, for the DFJSS problem, it is beneficial to prioritise increasing diversity when optimising the max-objective, while focusing more on convergence when optimising the mean-objective. This is because of the characteristic of these two types of objectives in DFJSS: 1) The max-objective is more sensitive to outliers, as it focuses on optimising the worst-case job/machine. If there are occasional jobs or machines that have significantly higher processing times, the max-objective may prioritise reducing the impact of those outliers, potentially at the expense of the majority of jobs. Consequently, there is a higher risk of getting trapped in

local optimal when handling the max-objective. 2) The mean-objective, on the other hand, is less sensitive to outliers and extreme values. It aims for a balanced performance across all jobs and machines. By converging towards solutions that distribute the workload evenly, the algorithm can ensure that no particular job or machine is significantly overburdened. This leads to improved overall system performance and minimises potential bottlenecks or delays. In summary, when optimising the max-objective, diversity is crucial to explore the search space effectively and avoid being stuck in local optimal. Conversely, when optimising the mean-objective, convergence is an important consideration to improve the average performance and achieve a balanced workload distribution.

4.5.5 Training Time

Different from other chapters in this thesis, the evaluation process in this chapter involves individuals and ensembles which might influence the training time. In this case, this chapter conducts training time analyses.

Table 4.6: The mean (standard deviation) training time (in minutes) of 30 independent runs of GP and EGP^e methods for 6 scenarios.

Scenario	GP	EGP ^e
<Fmax, 0.85>	103.84(10.91)	94.22(11.64)(↑)
<Fmax, 0.95>	109.21(11.87)	99.88(9.83)(↑)
<Tmax, 0.85>	103.24(11.89)	91.52(10.45)(↑)
<Tmax, 0.95>	109.91(13.08)	98.46(14.61)(↑)
<WTmean, 0.85>	108.80(17.29)	102.85(21.36)(↑)
<WTmean, 0.95>	111.83(18.32)	103.68(12.52)(↑)

Table 4.6 presents the mean (standard deviation) training time (in minutes) of 30 independent runs for both GP and EGP^e methods across six scenarios. Notably, EGP^e exhibits significantly shorter training times in comparison to GP. This is attributed to our specific approach to limiting EGP^e to yield the same number of scheduling heuristic evaluations

throughout the entire training process. To elaborate, suppose GP performs 1000 scheduling heuristic evaluations. With our proposed EGP^e , considering 40 ensembles, each with 5 scheduling heuristics, the total evaluation count consists of 40 ensemble evaluations and 800 ($1000 - 40 \times 5 = 800$) scheduling heuristic evaluations. To ensure a stable scheduling environment, we leverage the first 1000 jobs for warm-up purposes in each evaluation. For evaluating an ensemble, all the 5 scheduling heuristics in the ensemble share one DFJSS instance, which means that they share the same 1000 warm-up jobs. Compared with evaluating each scheduling heuristic independently (each scheduling heuristic needs 1000 warm-up jobs), we save 160000 ($40 \times (5 - 1) \times 1000 = 160000$) job schedules for every generation. Given the consideration of a large-scale DFJSS problem, the evaluation step consumes the most time during training. Despite the additional time taken for ensemble construction and selection, it has a minimal impact compared to the reduction of job processing during the evaluation process. Thus, the proposed EGP^e effectively achieves superior performance within a shorter training time, further validating its effectiveness. More precisely, the proposed method saves the training time by 8.72% ($8.72\% = (\frac{|94.22-103.84|}{103.84} + \dots + \frac{|103.68-111.83|}{111.83})/6 * 100\%$) compared to classical GP. In conclusion, the results demonstrate that EGP^e can offer superior performance in significantly less time, highlighting both the effectiveness and efficiency of this proposed approach.

4.5.6 Structure Analyses of Elements in Ensemble

Studying the structure of elements in an ensemble can help users understand the principles of each element. This understanding can give users more confidence in using the evolved ensemble [142]. In this case, we consider the tree structures of routing rules in an evolved ensemble of a single run on the scenario $\langle F_{max}, 0.85 \rangle$ of EGP^e as an example.

Figure 4.15 illustrates the structures of routing rules, the training fitness

of the ensemble, and the training ensemble contribution of each rule. The overall training fitness of this ensemble is 1359.25, better than the training fitness of each individual element, which ranges from 1424.19 to 1505.83. Additionally, the ensemble contribution of each individual element ranges from 0.92 to 0.99. The tree structures of these five routing rules can be expressed in the following simplified expressions:

$$\begin{aligned}
R_1 &= \min(\min(NIQ, SL) + 2PT + TRANT, WIQ) + PT + \min(\max(NOR - PT, 1), PT - NIQ + \max(PT, 1)) + TRANT + \max(TRANT, WIQ) \\
R_2 &= \min(\min(WIQ, PT - NOR + \min(SL, WKR)) + TRANT, WIQ) + PT + \min(\max(TRANT, WIQ), \frac{NOR - NIQ}{TRANT}) + \max(TRANT, WIQ) \\
R_3 &= 2 \min(\min(NIQ, SL) + PT + TRANT, WIQ) + WIQ + TRANT \\
R_4 &= \min(\min(NIQ, SL) + PT + TRANT, WIQ) + PT + \min(\frac{\max(\min(WIQ, PT), \frac{NOR}{OWT})}{TRANT}, TIS - \frac{PT}{NIQ}) + \max(TRANT, WIQ) \\
R_5 &= \min(TIS, WIQ) + 2PT + TRANT - NIQ + 2 \max(TRANT, WIQ) + \frac{NOR}{OWT}
\end{aligned} \tag{4.8}$$

To provide specific details, as illustrated in Eq. (4.8), the routing rule R_1 is composed of 6 terminals (NIQ, SL, PT, TRANT, WIQ, and NOR), with PT being the most frequently utilised terminal (appearing 6 times). Following closely is TRANT, employed 3 times. Meanwhile, the routing rule R_2 consists of 7 terminals (WIQ, PT, NOR, SL, WKR, TRANT, and NIQ), with WIQ and TRANT being the predominant terminals (each occurring 4 times). Additionally, PT and NOR are used 2 times. As for routing rule R_3 , it involves 5 terminals (NIQ, SL, PT, TRANT, and WIQ). TRANT and WIQ are the terminals most frequently employed, each appearing 3 times, followed by NIQ, SL, and PT, each used 2 times. Moving on to routing rule R_4 , it encompasses 8 terminals (NIQ, SL, PT, TRANT, WIQ, NOR, OWT, and TIS). PT claims the highest frequency (occurring 4 times). Both TRANT and WIQ are employed 3 times. Lastly, the routing rule R_5 is combined with 7 terminals (TIS, WIQ, PT, TRANT, NIQ, NOR, and OWT). Within this combination, WIQ and TRANT lead the pack with

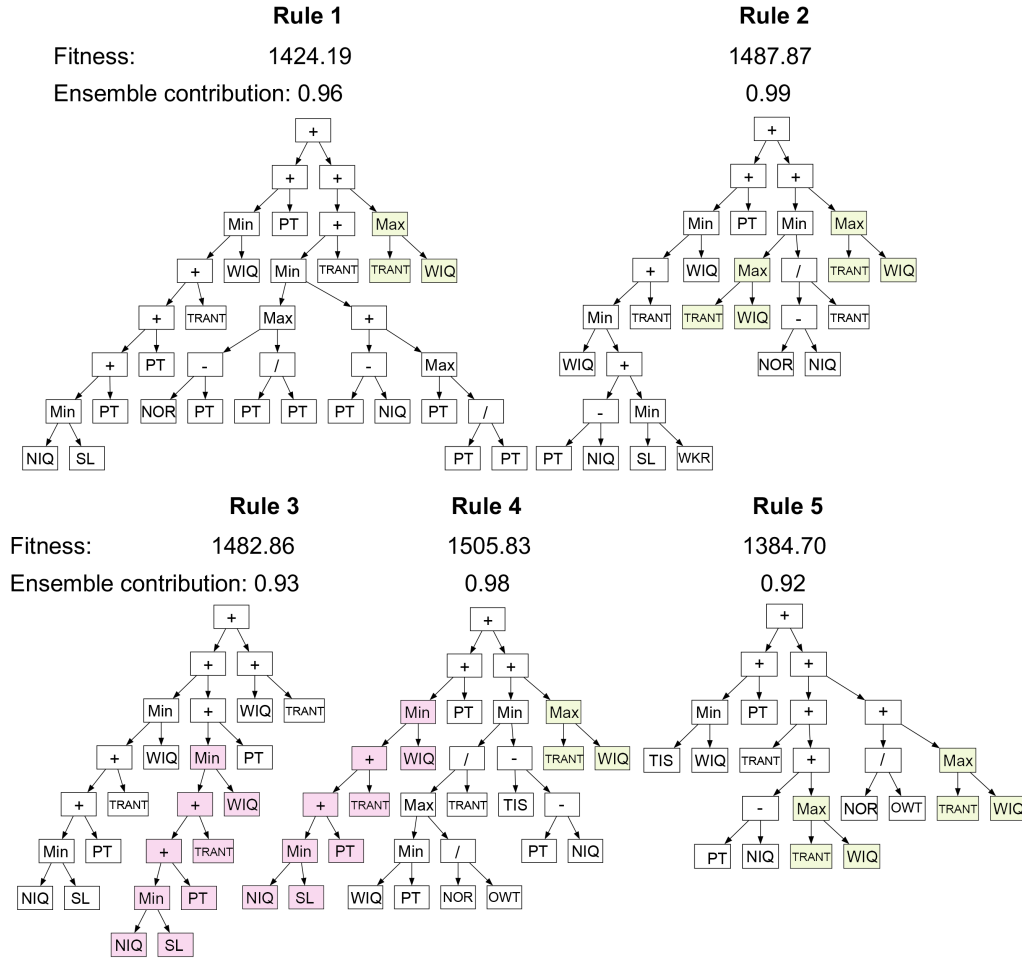


Figure 4.15: The tree structure of routing rule of an evolved ensemble with its training fitness and training ensemble contribution of a single run on scenario $\langle F_{\max}, 0.85 \rangle$ of EGP^e .

3 appearances. PT follows with 2 appearances. These rules behave differently (give different training fitness), but there are some similarities. We can see that there are some terminals (NIQ, PT, TRANT, and WIQ) used by all these five routing rules, also PT and TRANT play important roles (high use frequency) in the routing decision point, which is expected, as processing time (PT) and transportation time (TRANT) are two important

factors of the machine. Moreover, it is observed that some elements in the ensemble share the same subtrees. For example, R_1 , R_2 , R_4 , and R_5 all have the subtree $\max(TRANT, WIQ)$. R_3 and R_4 both have the subtree $\min(\min(NIQ, SL) + PT + TRANT, WIQ)$.

We can see that, the routing rules with some general subtrees can cover most decision points, but some different subtrees are required for some specific decision points, which means these different subtrees can help these rules complement with each other to make better overall decisions. Such a phenomenon inspires us to aim for learning a good rule that is generalisable to most decision points while also having subtrees that can play crucial roles in specific decision points.

4.6 Chapter Summary

This chapter proposes a novel EGP method named EGP^e , which allows the evolution of single individuals and ensembles together to solve the DFJSS problem. Extensive experiments and analyses demonstrate the effectiveness of our proposed method in terms of the evolved scheduling heuristic quality compared to existing recent popular GP methods. To be precise, the proposed method improves the test performance by 0.73% and saves the training time by 8.72% compared with the classic GP. Our proposed strategies, including ensemble construction and selection, and genetic operators considering both single individuals and ensembles, have also been verified to be effective in improving the performance of our EGP^e method.

Further analyses show that these strategies can generate high-quality single individuals and ensembles by preserving population diversity and supporting high ensemble contributions from ensemble elements. Moreover, the structural analyses of elements in the ensemble demonstrate that a promising ensemble contains elements with both shared subtrees and distinctive subtrees, allowing for effective complementarity and finally leading to improved overall joint decision-making capabilities. This com-

bination of the ensemble enables the ensemble to make reliable decisions for a wide range of decision points while also excelling in specific decision points. Overall, we believe that our findings contribute to the advancement of the field of GP and ensemble learning and have the potential for practical applications in real-world scheduling scenarios.

Chapters 3 mainly focus on evolving single scheduling heuristics to solve the DFJSS problems. This chapter focuses on developing an effective GP method with the ensemble learning technique that allows for evolving not only single scheduling heuristics to make individual decisions but also a group of scheduling heuristics to make joint decisions for solving the DFJSS problem. In the next chapter, we will further explore the utilisation of a group of scheduling heuristics evolved by GP via RL for solving the DFJSS problems.

Chapter 5

Collaborative Heuristic Generation and Selection by Genetic Programming and Reinforcement Learning

This chapter focuses on the development of a collaborative heuristic generation and selection method by integrating GP and RL for DFJSS. GP is leveraged to generate a diverse set of high-quality scheduling heuristics, while RL is employed to select the most suitable scheduling heuristics for different decision points in DFJSS.

5.1 Introduction

GP [24, 119] and RL [137] have demonstrated their effectiveness in automatically learning scheduling heuristics for dynamic scheduling problems. GP has been a successful approach for learning effective scheduling heuristics in DFJSS for a long time, and it remains dominant in this research field [262]. GP uses the evolutionary principles for learning

scheduling heuristics through an iterative, population-based, and stochastic evolutionary process [35]. Recently, RL has gained increasing attention in learning scheduling heuristics for dynamic scheduling problems. RL learns scheduling heuristics by interacting with the dynamic scheduling environment. Specifically, at each step, the RL agent chooses an action from the action space based on the current state, using the policy (scheduling heuristic) that maps states to actions. The agent then receives a reward and moves to the next state according to the state transition probability. For an episodic problem, the process repeats until the agent reaches a terminal state. After each episode, a discounted accumulated reward is obtained according to a discount factor. The agent's goal is to maximise the expectation of such long-term accumulated reward starting from any initial state [128]. GP and RL are mechanically dissimilar, although they both learn policies/scheduling heuristics to make decisions at decision points.

Currently, some RL-based approaches presume a fixed and constant number of machines/operations as actions at decision points and learn an end-to-end strategy to solve the JSS problems [101]. Consequently, these approaches face challenges when applied to DFJSS with varying numbers of machines/operations [101]. In response to this challenge, researchers have attempted to use indirect ways to overcome the difficulty of having different numbers of candidate machines/operations at different decision points. In particular, the utilisation of manually designed scheduling heuristics as actions in RL has been adopted for handling the varying number of machines/operations [36, 139]. In this approach, when RL selects a specific action (scheduling heuristic), the chosen scheduling heuristic is subsequently employed to make decisions regarding the candidate machines and operations. However, this kind of method still has limitations. Firstly, the manually designed scheduling heuristics often exhibit an average level of quality, thereby constraining the overall quality of high-level scheduling heuristics learned by RL. Secondly, the manual design process of a set of diverse scheduling heuristics is time-consuming

and requires a lot of domain knowledge from experts. GP [119] has been applied with notable success to address JSS problems [28]. Instead of relying on manually designed heuristics, GP employs an evolutionary process to iteratively learn and refine candidate scheduling heuristics over multiple generations. The key advantage of using GP for JSS lies in its ability to automatically explore a wide range of potential scheduling heuristics and adapt to the specific characteristics of the problem at hand with little domain knowledge [206]. Traditional GP methods typically focus on obtaining the best scheduling heuristic without emphasising the diversity within the population. Niching [246] has proven to be an effective strategy employed in GP to enhance its effectiveness [143]. This is achieved by increasing population diversity, fostering the presence of multiple diverse scheduling heuristics within the population. Leveraging the strengths of RL, GP, and the niching strategy, there is potential to explore the effectiveness of a hybrid method combining these methods for achieving enhanced scheduling results in DFJSS.

5.1.1 Chapter Goals

The goal of this chapter is to *investigate a hybrid two-stage GP and RL method enabling intelligent heuristics generation and selection across diverse decision points to enhance both GP and RL for solving the DFJSS problems effectively*. To be specific, the objectives of this chapter are as follows.

1. Propose a new two-stage learning method that integrates the benefits of GP and RL to learn intelligent scheduling agents to address the challenges of the DFJSS problems.
2. Propose a niching GP for the first stage, aiming to evolve a diverse set of heuristics as actions for RL. This approach offers several advantages: (1) it reduces dependency on domain-specific knowledge compared to manual heuristic design; (2) it yields actions with superior performance compared to manual heuristics; (3) it offers a vari-

ety of heuristics as actions, capable of addressing diverse decision-making scenarios.

3. Employ the RL method for the second stage that uses the learned high-quality and diverse scheduling heuristics by the niching GP as actions to learn intelligent scheduling agents for DFJSS.
4. Verify whether the proposed method outperforms the baseline RL and GP method, as well as widely used manually designed scheduling heuristics. Conduct further analyses to identify the factors contributing to the effectiveness of the proposed method.

5.1.2 Chapter Organisation

The subsequent sections of this chapter are organised as follows. Section 5.2 describes the proposed algorithm. The experimental design and results are detailed in Sections 5.3 and 5.4, respectively. Additional analyses are presented in Section 5.5. Lastly, Section 5.6 offers conclusions for this chapter.

5.2 Proposed Algorithm

We first conducted a comparison between a typical GP method and a typical RL method for DFJSS problems, as described in our publication [239]. After identifying the respective advantages of these two methods, we then proposed a hybrid GP and RL method, which is detailed as follows.

5.2.1 Overall Framework

The overall framework of the proposed algorithm is shown in Figure 5.1. The framework is denoted as niching GP-assisted DRL (NichGP-DRL). In the first stage, a niching GP (NichGP) method is presented to

autonomously learn a diverse set of high-quality scheduling heuristics. Subsequently, in the second stage, the learned sequencing rules by the NichGP are utilised as actions for the DRL method to adaptively select the most appropriate learned heuristic at different decision points. The subsequent sections provide detailed insights into the state features, the proposed NichGP method, and the DRL method.

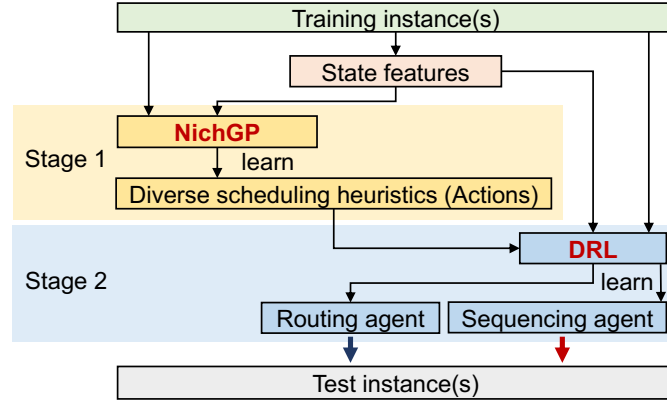


Figure 5.1: The overall framework of the proposed NichGPDRL method.

5.2.2 State Features

The state features utilised for both NichGP and DRL are derived from the following characterisations. These features remain consistent with the baseline DRL in [136] to ensure a fair comparison. Most of them correspond to the terminals typically used by GP for DFJSS, as discussed in previous chapters, while some are new additions based on the baseline DRL in [136]. For your convenience, this chapter lists the state features as follows. For further details, please refer to [136].

1. $PT_{j,i,m}(t)$: The processing time of the operation $O_{j,i}$ on the machine Ω_m at time t .

2. $WKR_j(t)$: The work remaining, representing the total processing time of the job J_j for the remaining operations at time t .
3. $CR_j(t)$: The completion rate, denoting the percentages of completed operations among all the operations of the job J_j at time t .
4. $TTD_j(t)$: The time until due, meaning the remaining time of the job J_j until the due date at time t .
5. $SLACK_j(t)$: The slack of the job J_j at time t , $SLACK_j(t) = t_j^{due} - t - WKR_j(t)$.
6. $WIQ_m(t)$: The remaining work (total processing time of all the operations) in the waiting queue of machine Ω_m at time t .
7. $NIQ_m(t)$: The number of operations in the waiting queue of machine Ω_m at time t .
8. $MRT_m(t)$: The ready time of machine Ω_m at time t , i.e., when machine becomes idle.
9. $MWT_m(t)$: The waiting time of machine Ω_m at time t , $MWT_m(t) = t - MRT_m(t)$.
10. $MBT_m(t)$: The busy time, denoting the total working time of machine M_m at time t .
11. $NPT_{j,i+1}(t)$: The median of the processing time for the next operation $O_{j,i+1}$ at time t .
12. $NOR_j(t)$: The number of the remaining operations of the job J_j at time t .
13. $OWT_{j,i}(t)$: The waiting time of the operation $O_{j,i}$ at time t , $OWT_{j,i}(t) = t - ORT_{j,i}(t)$, where $ORT_{j,i}(t)$ denotes the ready time of the operation $O_{j,i}$, which denotes the time of an operation arrives at the queue of the machine.

14. t : The current time of the scheduling system.
15. $TIS_j(t)$: The time that the job J_j has been in the scheduling system at time t , $TIS_j(t) = t - t_j^{arr}$.

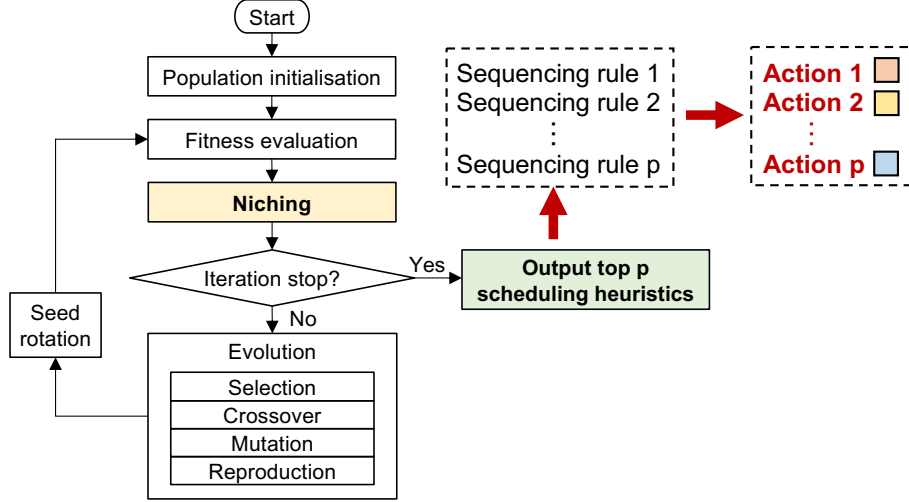


Figure 5.2: The flowchart of the GP method of Stage 1.

5.2.3 Stage 1: Niching GP training

Figure 5.2 gives the flowchart of NichGP. Different from the traditional GP method, NichGP uses a niching strategy [143] after fitness evaluation to remove duplicated and poor individuals from the population. In this case, NichGP aims to achieve the coexistence of multiple high-quality and diverse scheduling heuristics in the population. In DFJSS, phenotypic diversity is more meaningful than genotypic diversity [240]. In this chapter, phenotypic diversity is considered to manage a niche. We adopt the phenotypic characterisation (PC) [93, 143] to measure the phenotypic diversity. The PC is defined as a vector of values. Each value within the vector signifies the rank by the reference rule assigned to the candidate

machine/operation where the calculated rule designates the highest priority at a given decision point. This chapter uses the sequencing rule and the routing rule from the best individual at each generation as the reference sequencing rule and reference routing rule, respectively. To process a DFJSS instance, typically comprising thousands of decision points, we adopt a computational efficiency strategy by focusing on 20 sequencing decision points and an equivalent number of routing decision points. Therefore, a PC of a scheduling heuristic contains 40 values. To be specific, an unseen instance is used to get all the decisions which involves 7 candidate machines/operations. Then we shuffle the decisions and randomly get 20 sequencing decision points and 20 routing decision points. The PC for an individual involves a combination of sequencing and routing decisions. A detailed description of the PC has been given in Chapter 6.

The pseudo-code of NichGP is detailed in Algorithm 7. NichGP computes the PC, denoted as pc_i , for each individual ind_i within the population at every generation (see line 5). Subsequently, a clearing strategy [246] is applied, penalising individuals within a niche that show poor performance by assigning them an infinite fitness value (see line 6) [236]. The clearing strategy takes into account two crucial parameters: the *radius* δ of each niche, signifying the PC distance between niches, and the *capacity* κ of each niche, representing the number of high-quality individuals to be retained in the niche. In this chapter, we use Euclidean distance to calculate the distance between PCs. Notably, in contrast to traditional GP, NichGP employs the clearing strategy each time when the fitness evaluation process is conducted. Moreover, top p individuals are output as the actions for DRL. It is important to highlight that, while DRL only requires sequencing rules since it can learn an end-to-end routing agent for the DFJSS problem considered [136], our method involves evolving both routing and sequencing rules simultaneously using NichGP. The reason for this lies in our findings: when we experimented by keeping the routing rule fixed as a manual rule and evolving only the sequencing rule, we

Algorithm 7: The pseudo-code of NichGP.

Input: Population size: N ; Generations: G .
Output: A list of top p individuals: $\Delta = [ind_1, \dots, ind_p]$.

```

1 Initialise population  $pop$  with  $N$  individuals;
2  $g \leftarrow 0$ ;
3 while  $g < G$  do
4   Fitness evaluation for each individual  $ind_i$  in  $pop$ ;
   // Calculate PC of each individual
5    $pop \leftarrow calculatePC(pop)$ ;
   // Use clearing strategy to penalize individuals within
   // a niche that show poor performance
6    $Clearing(pop)$ ;
7   Selection;
8   Crossover/Mutation/Reproduction;
9    $g \leftarrow g + 1$ ;
10 end
11  $\Delta \leftarrow \emptyset$ ;
12  $i \leftarrow 0$ ;
13 while  $i < p$  do
14    $\Delta \leftarrow \Delta \cup ind_i$ ;
15 end
16 return  $\Delta$ ;
```

observed that the performance of the sequencing rules is constrained by the fixed routing rule, which will affect the effectiveness of the proposed method.

5.2.4 Stage 2: DRL training

We adopt the DRL framework with a DQN presented in [136]. However, our method differs in that we replace the actions used to train the sequencing agent with those learned through the proposed NichGP. The specifics of the DRL framework are presented as follows. Markov decision process (MDP) [65] is used to model the process that the DRL for solving the DFJSS problem. The MDP is a fundamental framework in RL that math-

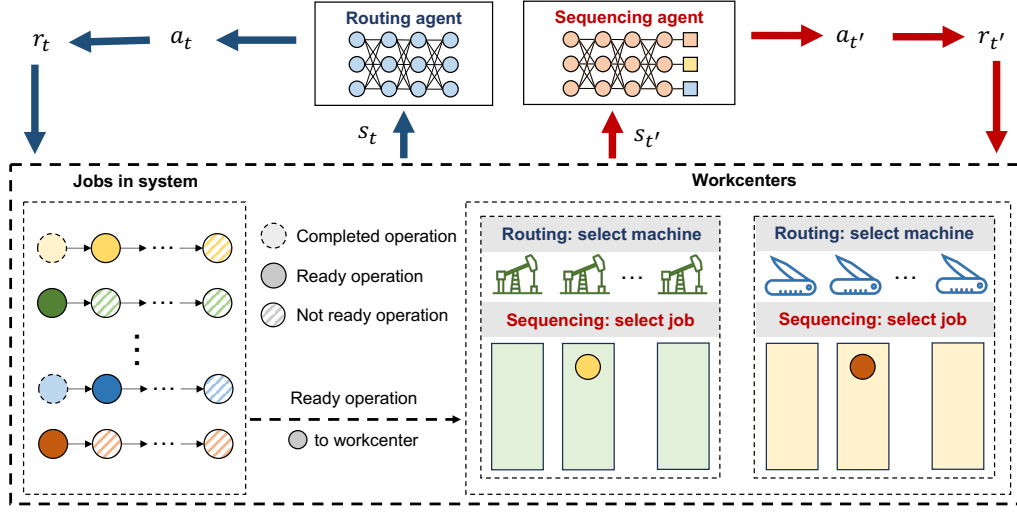


Figure 5.3: The flowchart of the DRL method of Stage 2.

ematically formalises decision-making in an environment where an agent interacts to achieve a goal. The Markov property plays an important role in utilising DQN effectively. It assumes that the future state depends only on the current state and the action taken, not the entire history of previous states and actions. Many complex problems (e.g., DFJSS) might not strictly adhere to the Markov property. However, by carefully designing the state representation, it is able to capture the essential information about the environment's history within the current state. This allows us to leverage DQNs even in situations with some degree of dependency on past actions. The MDP has a 5-element tuple representation: $\langle S, A, P, \gamma, R \rangle$. S represents the state space, which is the set of all possible situations or configurations that the environment can be in. A denotes the action space, which is the set of all possible actions that the agent can take. P is the transition probabilities, the probabilities associated with transitioning from one state (s_t) to another (s_{t+1}) after taking a specific action (a_t). It characterises the dynamics of the environment. R means the reward function, that specifies the immediate reward (r_t) the agent receives after transitioning. It is re-

lated to the goal or objective of the agent. γ represents the discount factor which is a parameter between 0 and 1 that discounts future rewards. It helps in balancing the trade-off between immediate and future rewards.

Figure 5.3 illustrates the flowchart of the DRL method. To mitigate the exponential growth in the state-action space during multi-agent DRL training [13], the sequencing agent and the routing agent are trained separately. This separation is employed to manage the coordination of exploration efforts, striking a balance between local and global exploration trade-offs [13]. Additionally, during the training of the sequencing agent, the routing rule is kept fixed as the earliest available machine, and the sequencing rules derived from the learned scheduling heuristics are utilised as actions. When training the routing agent, the sequencing rule is set as the First-in-First-out operation, and the routing agent takes end-to-end training, enabling it to make direct decisions among candidate machines. During the training process, the routing/sequencing agent observes the current state (s_t) of the environment and then selects an action (a_t) based on its policy (π). The environment transitions to a new state (s_{t+1}) according to the transition probabilities (P). The agent receives a reward (r_t) from the reward function. The agent updates its knowledge (Q function and policy) based on the observed state, action, reward, and the resulting state. The process repeats over multiple time steps as the agent learns to maximise cumulative rewards.

In this chapter, a double DQN architecture is used to train the agent. Double DQN is an enhancement of the traditional DQN, designed to address a common issue known as overestimation bias in Q-learning [186]. It is developed to improve the stability and performance of DQN algorithms. Double DQN uses two separate neural networks: one for action selection (referred to as the action network) and another for Q-value evaluation (referred to as the target network). The action-selection network is responsible for determining the best action, while the target network is used to estimate the Q-value of that action.

The ultimate goal of the agent is to find an optimal policy that maximises the expected cumulative reward over time in the given environment. This is typically achieved through the DRL method that iteratively improves the agent's policy based on its experiences. These experiences play a central role in the learning process. By exploring different actions in various states, the agent can gather information about the environment, which is essential for making informed decisions. The proposed DRL uses experience replay, where past experiences are stored in memory and randomly replayed during learning. This helps the agent learn from a diverse set of experiences and mitigates the impact of correlated sequential data. About the action representation, the routing agent's actions directly correspond to the selected machine within the workcenter, given the fixed number of machines in this study. The sequencing agent's actions entail the selection of a NichGP-learned sequencing rule. This approach addresses the challenge posed by a changing queue, making the direct selection of operations impractical. More details about the reward function and other processes can be found in [136].

5.3 Experimental Design

5.3.1 Dataset

This chapter explores a dynamic production system with continuous job arrivals. The dataset employed in this chapter differs from those in other chapters of this thesis and aligns with those specified in [136]. The reason for adopting the dataset from [136] is to enable a direct comparison between the proposed NichGP-assisted DRL method and the original DRL method presented in [136]. This way ensures fairness in the comparison and helps mitigate potential issues that may arise from transitioning an algorithm from its original dataset to a new one. To assess the performance of the proposed method and comparison methods, we examine four sce-

narios based on the following three critical factors:

1. **Job Arrival Rate / System Utilisation level:** the job arrival rate is closely linked to the system's utilisation level, denoted as $E(u)$. This metric can be calculated as follows:

$$E(u) = \frac{\mathbb{E}(t) \div \frac{k}{w}}{\mathbb{E}(in)} \times 100\% = \frac{\mathbb{E}(t) \times w}{\mathbb{E}(in) \times k} \times 100\%$$

where k and w represent the number of machines and workcenters, respectively. $\mathbb{E}(t)$ stands for the expected processing time of all operations across all machines, and $\mathbb{E}(in)$ represents the expected time interval between job arrivals. In this study, we assume a 90% utilisation level ($E(u)$) to simulate a busy production environment [136]. Additionally, we assume that the time interval (X) between successive job arrivals follows an exponential distribution [234]: $X \sim \text{Exp}(\mathbb{E}(in))$.

2. **Heterogeneity of Processing Time:** the processing time ($t_{j,i,m}^{pro}$) for each operation ($O_{j,i}$) on machine Ω_m is randomly sampled from a uniform distribution ($U[L_p, H_p]$), with L_p and H_p denoting the lower and upper limits of the processing time, respectively. For different scenarios, we consider different average processing times:

$$\begin{aligned} & \text{[]High heterogeneity: } t_{j,i,m}^{pro} \sim U[5, 25] \quad \text{Low heterogeneity:} \\ & t_{j,i,m}^{pro} \sim U[10, 20] \end{aligned}$$

3. **Due Date Tightness:** the due date (t_j^{due}) for each job (J_j) is assigned based on its expected total processing time and the due date factor (α_j). This chapter considers two types of due date tightness ranges ($U[L_d, H_d]$):

- High tension: $\alpha_j \sim U[1, 2]$
- Low tension: $\alpha_j \sim U[1, 3]$

The due date is calculated as follows:

$$t_j^{due} = t_j^{arr} + \alpha_j \sum_{i=1}^{q_j} \left(\frac{\sum_{m=1}^{k_{j,i}} t_{j,i,m}^{pro}}{k_{j,i}} \right), \alpha_j \sim U[L_d, H_d]$$

Based on the above descriptions, the four scenarios used to evaluate the performance are as follows:

- **HH**: the processing time exhibits high heterogeneity ($[5, 25]$), and high tension in due dates ($[1, 2]$);
- **HL**: the processing time exhibits high heterogeneity ($[5, 25]$), and low tension in due dates ($[1, 3]$);
- **LH**: the processing time has low heterogeneity ($[10, 20]$), and high tension in due dates ($[1, 2]$);
- **LL**: the processing time has low heterogeneity ($[10, 20]$), and low tension in due dates ($[1, 3]$).

Furthermore, the shop floor is equipped with 3 workcenters, each of which is equipped with 2 machines. Each instance of the simulation covers a production period of 1000 time units, during which approximately 124 jobs arrive on the shop floor.

5.3.2 Parameter Setting

The features described in Section 5.2.2 are used by both the NichGP and DRL. NichGP employs a function set $\{+, -, \times, /, \max, \min\}$, where $/$ is protected, returning 1 in case of division by 0. Additional parameters for NichGP are presented in Table 5.1. The parameter settings were carefully chosen to ensure a fair comparison with the baseline DRL method. The NichGP method is implemented in Python using the DEAP package [51]. With the advantage of parallel evaluation, the multiprocessing package [10] in Python is employed to speed up NichGP's training process. The

DRL implementation, utilising PyTorch [178] in Python, is detailed in Table 5.2 [136], encompassing parameters and network structures.

Table 5.1: The parameter settings of the proposed NichGP method.

Parameter	Value
Population size	200
Number of generations	50
Instances in each generation	2
Method for initialising population	Ramped-half-and-half
Initial minimum/maximum depth	2 / 6
Elitism	10
Maximal depth	8
Crossover rate	0.80
Mutation rate	0.15
Reproduction rate	0.05
Terminal/non-terminal selection rate	10% / 90%
Parent selection	Tournament selection with size 4
Output as actions for DRL	Top 4 individuals

Table 5.2: The parameter configuration of the DRL method.

Parameter	Routing	Sequencing
Exploration rate (ϵ)	0.3 decays to 0.1	0.3 decays to 0.1
Discount factor (γ)	0.8	0.8
Learning rate	0.01 decays to 0.001	0.01 decays to 0.001
Minibatch size	128	64
Replay memory size	512	256
Input layer size	9	25
Output layer size	2	4
Hidden layer size	$16 \times 16 \times 16 \times 8 \times 8$	$48 \times 36 \times 36 \times 24 \times 24 \times 12$
Channels	1	6

5.3.3 Comparison Design

To verify the effectiveness of our proposed NichGPDRM method, we compare our method with the following methods. This chapter incorporates four manually designed routing rules and four manually designed sequencing rules which are widely used in industry [139, 124]. These are combined to create 16 scheduling heuristics, forming the basis for comparison with the learned scheduling heuristics from NichGP and DRL methods. This comparative analysis with manually designed rules offers an intuitive perspective on the effectiveness of the proposed NichGPDRM method, highlighting their strong generalisation capabilities. The manually designed routing and sequencing rules taken into comparison are listed as follows.

Routing rules:

1. Earliest completion time (ECT): gives the machine that has the smallest sum of available time and remaining processing time the highest priority;
2. Minimum execution time (MET): gives the machine that has the minimum execution time the highest priority;
3. Earliest available (EA): gives the machine that has the earliest available time the highest priority;
4. Least work in the queue (LWIQ): gives the machine that has the least work remaining (total processing time) in its queue the highest priority.

Sequencing rules:

1. Shortest processing time (SPT): gives the operation that has the shortest processing time the highest priority;
2. Earliest due date (EDD): gives the operation whose job has the earliest due date the highest priority;

3. Least work remaining (LWR): gives the operation whose job has the least work remaining (processing time) the highest priority;
4. First-in-first-out (FIFO): gives the operation that arrives the first the highest priority.

In addition, the GP-assisted DRL (GPDRL) is used as a baseline to verify the effectiveness of the proposed NichGP method. Moreover, the method that NichGPDRL without DRL training is employed (NichGP#). For NichGP#, the sequencing rule remains fixed as one of the candidate rules (actions), serving to validate the efficacy of the DRL learning process. For each method, 30 independent runs are conducted to train 30 scheduling heuristics. In both NichGP (stage 1) and DRL (stage 2), the training process involves 100 instances. In the case of NichGP, two instances are applied per generation, and the top 4 scheduling heuristics, learned over 50 generations (totaling 100 instances), are returned. For DRL, the learned sequencing and routing agents are obtained after completing the 100 training instances. After training, we evaluate the objective values generated by the learned scheduling agents in 30 independent runs across 100 unseen instances for each of the four scenarios.

1. 16 widely used manual scheduling heuristics;
2. **DRL** [136]: the routing agent is trained to directly select from machines/operations while the sequencing agent is trained to select from manual sequencing rules;
3. **GPDRL**: the routing agent is trained to directly select from machines/operations while the sequencing agent is trained to select from the learned sequencing rules by the original GP;
4. **NichGP#** ($\# \in [1, 2, 3, 4]$): the routing agent is trained through DRL [136] using an end-to-end way, directly selecting from machines/operations. Meanwhile, the sequencing agent remains fixed as one of the rules learned by the proposed NichGP;

5. **NichGPDRL** (ours): the routing agent is trained to directly select from machines/operations while the sequencing agent is trained to select from the learned sequencing rules by the proposed NichGP.

By comparing with these methods, we can validate the effectiveness of the proposed NichGPDRL method, assess the performance of learned sequencing rules by GP methods in comparison to manually designed ones, and evaluate the efficacy of the proposed niching strategy in learning diverse and high-quality sequencing rules, as opposed to the traditional GP method.

Table 5.3: The mean and standard deviation training performance of 30 independent runs of the proposed NichGPDRL method with different δ values.

δ	HH	HL	LH	LL	Rank
0	537.40(333.24)	173.40(194.79)	1492.33(900.92)	636.03(627.94)	1.25
1	551.82(353.12)	172.60(184.53)	1519.75(885.75)	651.72(648.31)	1.75
2	561.35(344.87)	187.45(194.74)	1540.70(859.74)	664.25(656.11)	3
3	588.73(386.7)	194.70(202.84)	1594.92(937.7)	689.90(676.96)	4
4	616.37(367.32)	205.52(183.95)	1599.77(903.5)	695.33(693.09)	5
5	648.45(384.99)	225.05(257.65)	1667.58(933.76)	730.05(713.3)	6

5.4 Results and Discussions

5.4.1 Influence of the Radius Parameter in NichGP

The parameter radius, denoted as δ , plays a critical role in NichGP, representing the degree of dissimilarity between individuals within the NichGP. This dissimilarity level, in turn, can have an impact on the performance of the learned agent through RL. To explore this effect, we conduct experiments using six δ values: 0, 1, 2, 3, 4, and 5. Specifically, when $\delta = 0$, it

signifies that a distance larger than 0 between scheduling heuristics is considered acceptable, while other scheduling heuristics are penalised. The respective training performance of the proposed NichGPDRL method for each δ value is shown in Table 5.3. The results of the Friedman test [287], revealing a p-value of 0.002 (falling below the significance threshold of 0.05), indicate significant differences among different δ values. Considering the average rank of these methods across all four scenarios, a radius of 0 attains the highest rank with a value of 1.25. Additionally, we observe that as the radius value increases, the rank decreases. This phenomenon suggests that increasing the radius does not lead to performance improvement but rather degrades it. A radius of 0 yields the best performance. Consequently, for subsequent experiments and analyses, we use NichGPDRL to represent the proposed method with a radius of 0.

5.4.2 Test Performance

Table 5.4 gives the mean and standard deviation test performance of 30 independent runs of the proposed NichGPDRL methods and comparison methods. The Friedman test results, yielding a p-value of 4.28×10^{-8} (smaller than 0.05), indicate significant differences among the methods. Notably, all scheduling heuristics learned by hyper-heuristic methods (DRL, NichGP#, GPDRL, and the proposed NichGPDRL) outperform all the manually designed ones. Among the hyper-heuristic methods, DRL exhibits the worst performance. The proposed NichGPDRL obtains the best performance, with NichGP1 following closely as the second best. NichGP3 secures the third position, while NichGP2 takes the fourth position. NichGP4 and GPDRL closely follow in the fifth position.

To be more specific, we employ the Wilcoxon rank sum test [226] to compare the proposed NichGPDRL method with each comparison method across the four scenarios. The results, denoted by significantly better (\uparrow), worse (\downarrow), or statistically similar ($=$) compared to other meth-

Table 5.4: The mean and standard deviation test performance of 30 independent runs of the proposed NichGPDRL methods and comparison methods.

Algorithm	HH	HL	LH	LL	Rank
ECT+SPT	1014.29(0.00)(↑)	611.06(0.00)(↑)	2457.59(0.00)(↑)	1699.70(0.00)(↑)	9
ECT+EDD	1082.88(0.00)(↑)	441.37(0.00)(↑)	2424.66(0.00)(↑)	1265.08(0.00)(↑)	8
ECT+FIFO	1245.73(0.00)(↑)	705.08(0.00)(↑)	2748.60(0.00)(↑)	1727.35(0.00)(↑)	10.5
ECT+LWR	1305.73(0.00)(↑)	791.42(0.00)(↑)	2505.60(0.00)(↑)	1607.48(0.00)(↑)	10.5
MET+SPT	1437.83(0.00)(↑)	960.87(0.00)(↑)	4534.33(0.00)(↑)	3509.15(0.00)(↑)	16.5
MET+EDD	1719.43(0.00)(↑)	782.12(0.00)(↑)	4755.69(0.00)(↑)	3087.73(0.00)(↑)	16.25
MET+FIFO	1965.50(0.00)(↑)	1223.73(0.00)(↑)	5274.61(0.00)(↑)	3800.25(0.00)(↑)	18.5
MET+LWR	2028.35(0.00)(↑)	1345.42(0.00)(↑)	4966.10(0.00)(↑)	3693.70(0.00)(↑)	18.5
EA+SPT	3261.69(0.00)(↑)	2297.7(0.00)(↑)	3409.09(0.00)(↑)	2430.36(0.00)(↑)	15.25
EA+EDD	3824.05(0.00)(↑)	2235.12(0.00)(↑)	3403.57(0.00)(↑)	1901.17(0.00)(↑)	14.5
EA+FIFO	4254.86(0.00)(↑)	2871.10(0.00)(↑)	3930.47(0.00)(↑)	2617.09(0.00)(↑)	19.75
EA+LWR	3937.37(0.00)(↑)	2740.29(0.00)(↑)	3491.85(0.00)(↑)	2352.15(0.00)(↑)	16.75
LWIQ+SPT	3496.10(0.00)(↑)	2461.64(0.00)(↑)	3601.45(0.00)(↑)	2568.60(0.00)(↑)	17.25
LWIQ+EDD	4003.76(0.00)(↑)	2305.48(0.00)(↑)	3524.99(0.00)(↑)	2045.50(0.00)(↑)	16.5
LWIQ+FIFO	4468.28(0.00)(↑)	3022.68(0.00)(↑)	4064.07(0.00)(↑)	2720.61(0.00)(↑)	21
LWIQ+LWR	4219.96(0.00)(↑)	2945.80(0.00)(↑)	3688.77(0.00)(↑)	2505.47(0.00)(↑)	19
DRL	1003.30(43.95)(↑)	463.28(43.48)(↑)	2383.47(70.08)(↑)	1219.51(70.33)(↑)	7.25
NichGP1	929.08(54.08)(=)	385.49(34.95)(↑)	2255.22(83.52)(=)	1125.61(77.46)(=)	3
NichGP2	943.44(77.26)(=)	388.38(46.60)(=)	2285.84(95.21)(=)	1115.44(47.15)(=)	4
NichGP3	932.07(63.95)(=)	381.97(43.54)(=)	2273.90(91.91)(=)	1122.20(79.08)(=)	3.75
NichGP4	944.85(63.73)(=)	390.78(43.78)(↑)	2269.25(100.79)(=)	1119.19(53.11)(=)	4.25
GPDR	929.84(43.46)(=)	375.08(32.04)(=)	2299.29(88.82)(↑)	1211.13(64.51)(↑)	4.25
NichGPDRL	927.50(42.44)	372.18(19.13)	2263.42(70.21)	1121.14(75.55)	1.75

ods, are presented alongside the results of the comparison method in Table 5.4. Upon analysing these results, we observe that the proposed NichGPDRL significantly outperforms DRL and all manually designed scheduling heuristics across all four scenarios. When compared with NichGP#, the proposed NichGPDRL performs better than NichGP1 and NichGP4 on the scenario HL. When compared with GPDR, NichGPDRL performs significantly better than GPDR on two scenarios (LH and LL) while perform-

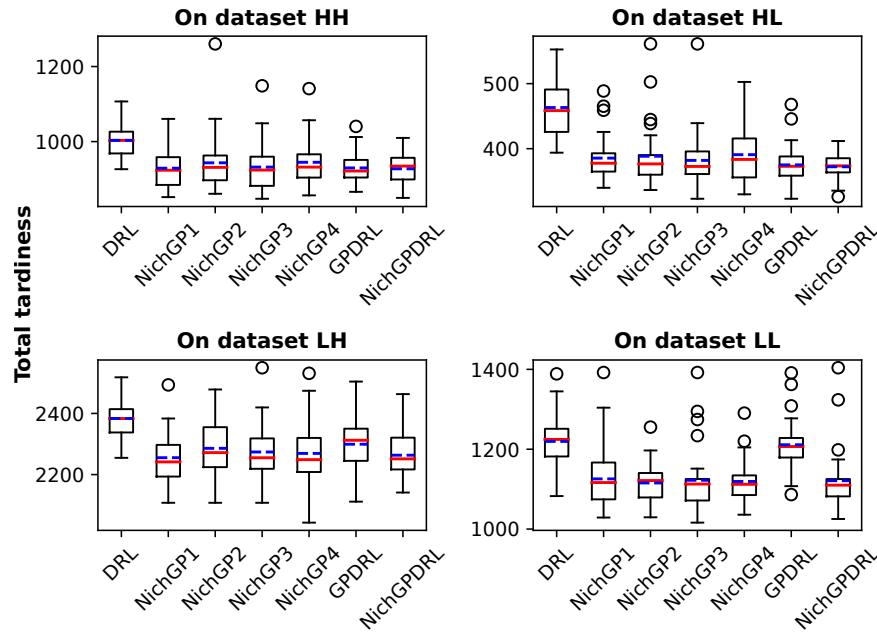


Figure 5.4: The box plots of the test performance of 30 independent runs of the proposed NichGPDRL, GPDRL, NichGP#, and DRL methods.

ing similarly with GPDRL on the other scenarios (HH and HL). Importantly, it never performs worse than the other methods in any other scenarios. These results validate the effectiveness of the proposed NichGPDRL method and highlight the effectiveness of the proposed niching strategy in aiding GP to generate high-quality and diverse actions for DRL compared to the original GP.

Figure 5.4 visualises the box plots of the test performance of 30 independent runs of the proposed NichGPDRL and other hyper-heuristic methods. The box plots highlight the significant advantages of the proposed algorithm compared to the DRL algorithm. It illustrates the benefits of combining GP and DRL over using DRL independently. The demonstrated advantages emphasise the effectiveness of the proposed NichGPDRL of integrating GP into DRL. Furthermore, the effectiveness of the proposed NichGP# is verified through these results.

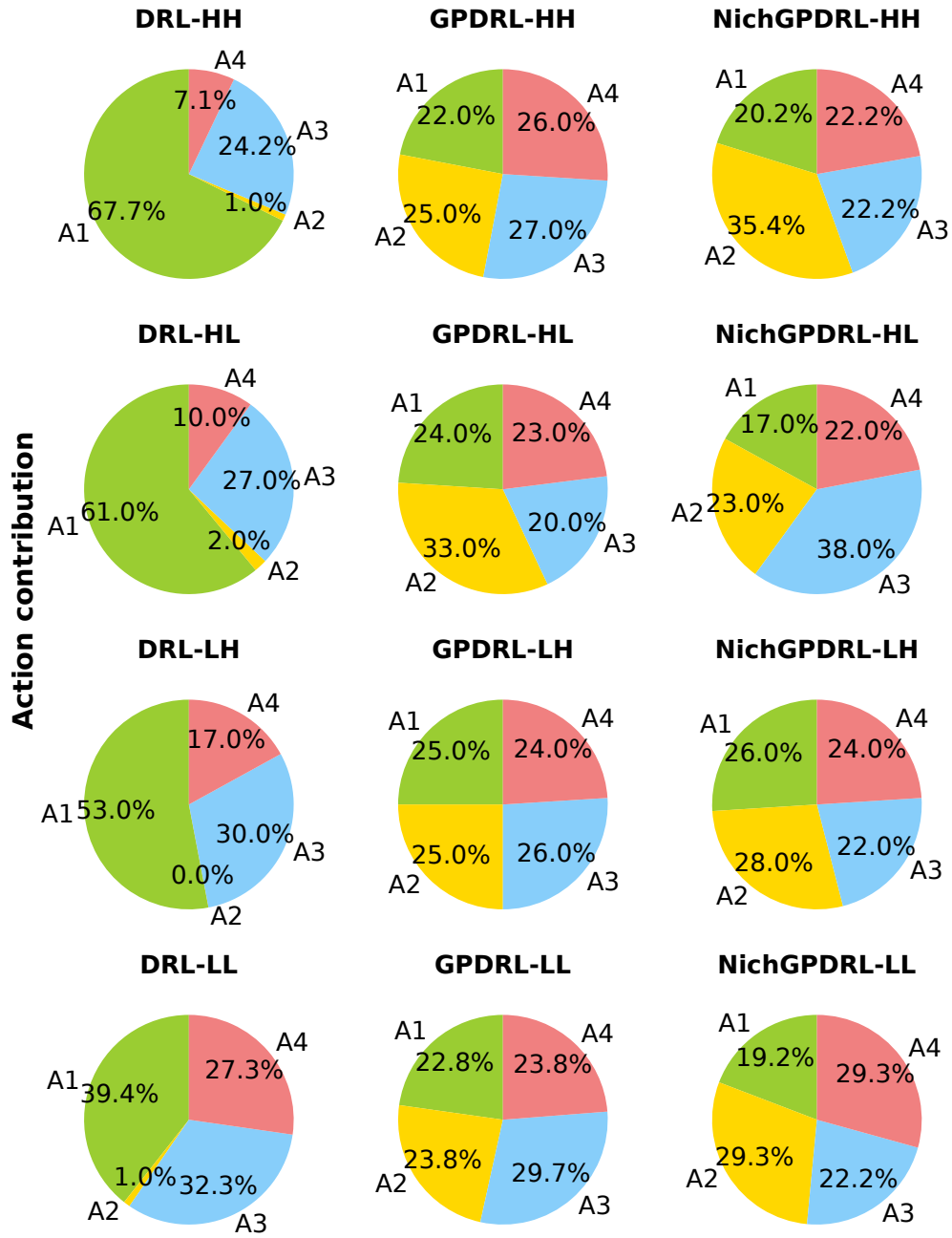


Figure 5.5: The pie plots of the **action contribution** of 30 independent runs of the DRL, GPDRL, and the proposed NichGPDRL methods on four scenarios.

5.5 Further Analyses

5.5.1 Action Contribution

In the learning process of RL, the selection process involves choosing from four scheduling heuristics (actions) at each decision point. This section focuses on the analysis of *action contribution*, defined as the percentage of times each action is utilised relative to the total number of decision points. The mean of action contributions is calculated across 30 independent runs for the DRL, GPDRL, and the proposed NichGPDL methods across four scenarios.

The results are presented in Figure 5.5 using pie plots. The pie plots reveal that, in the case of the DRL method, action 1 (A1, SPT) assumes a key role across all four scenarios, securing the highest percentage in each scenario, exceeding 50% in three of them. Subsequently, action 3 (A3, SLACK) secures the second-highest percentage. Action 4 (A4, CR) attains the third rank, while action 2 (A2, WIQ) ranks the lowest, constituting about only 1% on all four scenarios. In contrast to the DRL method, GPDRL, and NichGPDL methods exhibit similar action contributions across the four actions. This shows that the scheduling heuristics evolved by GP and NichGP can generate key rules that contribute to the overall performance.

5.5.2 Behaviour Difference between Heuristics

This chapter proposes a NichGP method to iteratively evolve a set of diverse scheduling heuristics for use as actions in RL. This section investigates the *behaviour difference* among scheduling heuristics evolved through the proposed NichGP. To ensure a fair comparison, we employ an unseen instance to generate 100 sequencing decisions. Subsequently, each scheduling heuristic evolved by NichGP and GP is evaluated on these 100 decisions, with the Hamming distance utilised to calculate the behavioural difference between each pair of sequencing rules (a total of 6 pairs for 4 se-

quencing rules). The average percentage of behavioural difference across all pairs is then computed as the behaviour difference ρ . The function to calculate the behaviour difference is as Eq. (5.1).

$$\rho = \frac{\sum_{i=1}^6 \text{Hamming}(\text{seq}_a, \text{seq}_b)}{6} \quad (5.1)$$

where, seq_a and seq_b represent different sequencing rules. Table 5.5 gives the mean and standard deviation behaviour difference of 30 independent runs of the proposed NichGPDRL and GPDRL methods. It can be seen that the proposed NichGPDRL shows significantly higher behaviour difference than the GPDRL on all the four scenarios. The NichGPDRL outperforms the GPDRL about more than 3 times on behaviour difference. This verifies the effectiveness of the proposed NichGP in learning diverse scheduling heuristics for DRL.

Table 5.5: The mean and standard deviation behaviour difference of 30 independent runs of the proposed NichGPDRL and GPDRL methods.

Scenario	GPDRL	NichGPDRL
HH	0.051(0.097)	0.161(0.066)(↑)
HL	0.054(0.058)	0.126(0.089)(↑)
LH	0.045(0.080)	0.205(0.074)(↑)
LL	0.045(0.058)	0.119(0.044)(↑)

5.5.3 Generalisation to More Complex Scenarios

The ability to generalise is a crucial metric for evaluating an algorithm. On one hand, an algorithm's generalisation ability can be assessed by examining the performance of its trained heuristics on unseen test instances at similar scales. In this context, as demonstrated in the comparison of test performances in Table 5.4, it is evident that the proposed NichGPDRL exhibits superior generalisation ability compared to DRL. On the other hand,

generalisation ability can be further validated by extending the application of trained heuristics to unseen test instances at more complex scales. To achieve this, scheduling heuristics learned by the proposed NichGP-DRL and comparison methods are tested on instances featuring a higher number of jobs and a large number of workcenters. Specifically, the test instances involve 248 jobs arriving over 2000 units of time and 620 jobs arriving over 5000 units of time. Similar to Table 5.4, 100 unseen instances are employed for testing. Furthermore, the test performance of 16 manually designed scheduling heuristics is presented as a baseline for comparison, to verify the effectiveness of the proposed method.

Table 5.6: The mean and standard deviation test performance of the DRL, the NichGPDRL, and comparison methods on eight scenarios with a large number of jobs arrival (2000 unit times).

Algorithm	HH	HL	LH	LL
ECT+SPT	2175.22(0.00)(↑)	1353.44(0.00)(↑)	5257.93(0.00)(↑)	3698.47(0.00)(↑)
ECT+EDD	2441.94(0.00)(↑)	1026.63(0.00)(↑)	5246.53(0.00)(↑)	2806.27(0.00)(↑)
ECT+LWR	2830.33(0.00)(↑)	1763.24(0.00)(↑)	5510.60(0.00)(↑)	3653.18(0.00)(↑)
ECT+FIFO	2730.64(0.00)(↑)	1588.24(0.00)(↑)	5938.43(0.00)(↑)	3806.03(0.00)(↑)
MET+SPT	3342.30(0.00)(↑)	2278.09(0.00)(↑)	10838.54(0.00)(↑)	8710.86(0.00)(↑)
MET+EDD	4088.80(0.00)(↑)	2065.01(0.00)(↑)	12083.85(0.00)(↑)	8248.97(0.00)(↑)
MET+LWR	4830.47(0.00)(↑)	3282.75(0.00)(↑)	12079.90(0.00)(↑)	9342.84(0.00)(↑)
MET+FIFO	4726.93(0.00)(↑)	3020.21(0.00)(↑)	12819.83(0.00)(↑)	9601.56(0.00)(↑)
EA+SPT	8602.50(0.00)(↑)	6419.72(0.00)(↑)	7761.20(0.00)(↑)	5715.40(0.00)(↑)
EA+EDD	10251.10(0.00)(↑)	6596.72(0.00)(↑)	8042.18(0.00)(↑)	4788.28(0.00)(↑)
EA+LWR	10526.01(0.00)(↑)	7767.38(0.00)(↑)	8125.91(0.00)(↑)	5670.46(0.00)(↑)
EA+FIFO	11028.04(0.00)(↑)	7884.14(0.00)(↑)	8895.89(0.00)(↑)	6108.21(0.00)(↑)
LWIQ+SPT	8955.59(0.00)(↑)	6648.55(0.00)(↑)	8133.55(0.00)(↑)	5960.71(0.00)(↑)
LWIQ+EDD	10569.78(0.00)(↑)	6836.88(0.00)(↑)	8335.29(0.00)(↑)	5063.08(0.00)(↑)
LWIQ+LWR	10975.34(0.00)(↑)	8123.35(0.00)(↑)	8442.37(0.00)(↑)	5864.28(0.00)(↑)
LWIQ+FIFO	11560.02(0.00)(↑)	8316.44(0.00)(↑)	9231.12(0.00)(↑)	6336.31(0.00)(↑)
DRL	2154.31(96.78)(↑)	1041.14(86.75)(↑)	4996.29(157.03)(↑)	2599.83(153.23)(↑)
NichGPDRL	2027.05(100.62)	882.36(45.47)	4813.97(234.03)	2415.20(147.26)

Tables 5.6, 5.7, 5.8, and 5.9 present the mean and standard deviation of test performances for the proposed NichGPDRL and comparison methods across two sets of scenarios: one characterised by a substantial number of job arrivals and the other featuring different numbers of workcenters. The “↑” next to the comparison method in these tables signifies that the results of NichGPDRL are significantly better than those of all the compared algorithms. These findings demonstrate that NichGPDRL outperforms DRL and all manually designed scheduling heuristics, even when directly applied to more complex instances without retraining. While DRL outperforms most manually designed scheduling heuristics in 16 scenarios, exceptions include ECT+EDD on scenario HL with 248 jobs, ECT+SPT on scenario HH with 620 jobs, ECT+EDD on scenario LL with six workcenters, each with two machines, and ECT+EDD on scenarios HL and LH with nine workcenters, each with two machines. These findings demonstrate that both NichGPDRL and DRL methods can be applied to more complex scenarios without retraining. Moreover, NichGPDRL exhibits superior generalisation ability compared to DRL as it provides even better performance.

5.5.4 Discussions on Enhancing GP capability

This chapter proposes a novel two-stage framework aimed at harnessing the strengths of both GP and RL to tackle the DFJSS problem. As a preliminary work to investigate the combination of these two methods, this chapter builds upon a baseline DRL method proposed in [136]. In [136], an end-to-end DRL method is proposed to learn a routing agent capable of directly selecting from candidate machines when encountering a routing decision point. Additionally, a heuristic selection DRL approach is presented to learn a sequencing agent responsible for selecting from manual sequencing rules to make further decisions among candidate operations. In this context, this chapter focuses on replacing the manual sequencing

Table 5.7: The mean and standard deviation test performance of the DRL, the NichGPDRL, and comparison methods on eight scenarios with a large number of jobs arrival (5000 unit times).

Algorithm	HH	HL	LH	LL
ECT+SPT	5670.90(0.00)(↑)	3540.29(0.00)(↑)	14346.88(0.00)(↑)	10342.80(0.00)(↑)
ECT+EDD	6479.19(0.00)(↑)	2924.44(0.00)(↑)	14474.19(0.00)(↑)	8065.42(0.00)(↑)
ECT+LWR	7733.09(0.00)(↑)	4849.61(0.00)(↑)	15235.86(0.00)(↑)	10315.41(0.00)(↑)
ECT+FIFO	7528.22(0.00)(↑)	4450.15(0.00)(↑)	16426.80(0.00)(↑)	10918.11(0.00)(↑)
MET+SPT	8741.83(0.00)(↑)	6000.12(0.00)(↑)	30705.00(0.00)(↑)	25227.22(0.00)(↑)
MET+EDD	10906.19(0.00)(↑)	5394.65(0.00)(↑)	34920.13(0.00)(↑)	25013.67(0.00)(↑)
MET+LWR	12775.60(0.00)(↑)	8787.98(0.00)(↑)	35324.91(0.00)(↑)	28118.48(0.00)(↑)
MET+FIFO	12569.91(0.00)(↑)	8111.35(0.00)(↑)	37149.28(0.00)(↑)	28687.94(0.00)(↑)
EA+SPT	25364.02(0.00)(↑)	19582.27(0.00)(↑)	22400.26(0.00)(↑)	17089.66(0.00)(↑)
EA+EDD	31222.59(0.00)(↑)	22163.05(0.00)(↑)	23034.26(0.00)(↑)	14708.75(0.00)(↑)
EA+LWR	33417.17(0.00)(↑)	25770.86(0.00)(↑)	23429.87(0.00)(↑)	16962.71(0.00)(↑)
EA+FIFO	33899.75(0.00)(↑)	25385.47(0.00)(↑)	25530.17(0.00)(↑)	18210.69(0.00)(↑)
LWIQ+SPT	26428.54(0.00)(↑)	20302.60(0.00)(↑)	23554.17(0.00)(↑)	17867.04(0.00)(↑)
LWIQ+EDD	31893.24(0.00)(↑)	22543.42(0.00)(↑)	24278.45(0.00)(↑)	15397.05(0.00)(↑)
LWIQ+LWR	33286.78(0.00)(↑)	25493.15(0.00)(↑)	24435.33(0.00)(↑)	17631.24(0.00)(↑)
LWIQ+FIFO	34631.34(0.00)(↑)	25898.14(0.00)(↑)	26808.94(0.00)(↑)	19191.36(0.00)(↑)
DRL	5781.12(253.85)(↑)	2834.20(214.81)(↑)	14065.40(448.60)(↑)	7773.84(431.70)(↑)
NichGPDRL	5386.88(231.38)	2397.40(144.05)	13534.79(609.15)	7333.81(578.69)

rules with sequencing rules learned by GP to enhance the effectiveness of the sequencing agent, while retaining the use of the end-to-end DRL for learning the routing agent. Based on the results and analyses presented, we demonstrate that the proposed NichGPDRL method significantly outperforms the baseline DRL method (i.e., $GP + RL > RL$), while achieving statistically similar performance to NichGP (which utilises an end-to-end routing agent and single GP-learned sequencing rule). However, we are unable to conclusively prove that the proposed NichGPDRL method outperforms the baseline GP method (i.e., $GP + RL > GP$). In future research, it would be intriguing to explore whether $GP + RL$ indeed outperforms

Table 5.8: The mean and standard deviation test performance of the DRL, the NichGP DRL, and comparison methods on eight scenarios with a large number of workcenters (6 workcenters).

Algorithm	HH	HL	LH	LL
ECT+SPT	1771.04(0.00)(↑)	988.88(0.00)(↑)	4323.62(0.00)(↑)	2646.83(0.00)(↑)
ECT+EDD	1780.71(0.00)(↑)	522.47(0.00)(↑)	3983.73(0.00)(↑)	1503.14(0.00)(↑)
ECT+LWR	2205.35(0.00)(↑)	1229.67(0.00)(↑)	4187.14(0.00)(↑)	2440.70(0.00)(↑)
ECT+FIFO	2244.21(0.00)(↑)	1183.49(0.00)(↑)	4860.78(0.00)(↑)	2749.25(0.00)(↑)
MET+SPT	2674.45(0.00)(↑)	1637.45(0.00)(↑)	7225.88(0.00)(↑)	5022.53(0.00)(↑)
MET+EDD	2803.78(0.00)(↑)	853.85(0.00)(↑)	7005.63(0.00)(↑)	3453.68(0.00)(↑)
MET+LWR	3499.82(0.00)(↑)	2085.59(0.00)(↑)	7596.67(0.00)(↑)	5030.36(0.00)(↑)
MET+FIFO	3668.17(0.00)(↑)	2067.90(0.00)(↑)	8573.12(0.00)(↑)	5477.07(0.00)(↑)
EA+SPT	6080.39(0.00)(↑)	3915.84(0.00)(↑)	6100.94(0.00)(↑)	3922.60(0.00)(↑)
EA+EDD	6625.08(0.00)(↑)	3117.37(0.00)(↑)	5667.42(0.00)(↑)	2458.70(0.00)(↑)
EA+LWR	6573.10(0.00)(↑)	4097.93(0.00)(↑)	5811.79(0.00)(↑)	3534.29(0.00)(↑)
EA+FIFO	7772.72(0.00)(↑)	4799.38(0.00)(↑)	6756.52(0.00)(↑)	4019.94(0.00)(↑)
LWIQ+SPT	6253.02(0.00)(↑)	3993.08(0.00)(↑)	6205.48(0.00)(↑)	3972.28(0.00)(↑)
LWIQ+EDD	6862.33(0.00)(↑)	3397.43(0.00)(↑)	5896.02(0.00)(↑)	2732.58(0.00)(↑)
LWIQ+LWR	7100.61(0.00)(↑)	4486.30(0.00)(↑)	5999.24(0.00)(↑)	3637.71(0.00)(↑)
LWIQ+FIFO	8034.78(0.00)(↑)	4975.42(0.00)(↑)	6958.58(0.00)(↑)	4174.30(0.00)(↑)
DRL	1597.57(97.59)(↑)	570.07(103.14)(↑)	3841.77(142.62)(↑)	1401.16(165.89)(↑)
NichGP DRL	1460.45(72.37)	375.57(23.01)	3709.01(137.02)	1244.96(64.20)

GP by further investigating a heuristic selection DRL for learning routing agents or investigating different combination methods of GP and RL.

5.5.5 Structure Analysis of Sequencing Rules

In this section, we analyse the structure of sequencing rules evolved by NichGP and used as actions for DRL. We randomly select a run and the structures of the 4 sequencing rules are shown in Figures 5.6, 5.7, 5.8, and 5.9.

For the sequencing rule 1 in Figure 5.6, it comprises four terminals: PT, SLACK, NIQ, and WKR. Notably, SLACK emerges as the predominant ter-

Table 5.9: The mean and standard deviation test performance of the DRL, the NichGPDRL, and comparison methods on eight scenarios with a large number of workcenters (9 workcenters).

Algorithm	HH	HL	LH	LL
ECT+SPT	3171.25(0.00)(↑)	1742.07(0.00)(↑)	7411.42(0.00)(↑)	4459.58(0.00)(↑)
ECT+EDD	2918.00(0.00)(↑)	712.32(0.00)(↑)	6182.24(0.00)(↑)	2045.61(0.00)(↑)
ECT+LWR	3603.28(0.00)(↑)	1942.15(0.00)(↑)	6829.04(0.00)(↑)	3862.74(0.00)(↑)
ECT+FIFO	4209.50(0.00)(↑)	2199.16(0.00)(↑)	8202.33(0.00)(↑)	4522.05(0.00)(↑)
MET+SPT	4191.48(0.00)(↑)	2422.35(0.00)(↑)	11246.05(0.00)(↑)	7509.85(0.00)(↑)
MET+EDD	4084.71(0.00)(↑)	935.13(0.00)(↑)	10013.13(0.00)(↑)	4300.95(0.00)(↑)
MET+LWR	5312.86(0.00)(↑)	3000.78(0.00)(↑)	11339.11(0.00)(↑)	7204.76(0.00)(↑)
MET+FIFO	5872.18(0.00)(↑)	3139.97(0.00)(↑)	12896.23(0.00)(↑)	7883.16(0.00)(↑)
EA+SPT	9681.76(0.00)(↑)	6014.37(0.00)(↑)	9606.73(0.00)(↑)	5984.38(0.00)(↑)
EA+EDD	9566.29(0.00)(↑)	4112.08(0.00)(↑)	8376.61(0.00)(↑)	3331.55(0.00)(↑)
EA+LWR	10398.22(0.00)(↑)	6304.28(0.00)(↑)	9149.84(0.00)(↑)	5372.83(0.00)(↑)
EA+FIFO	12197.22(0.00)(↑)	7209.15(0.00)(↑)	10862.16(0.00)(↑)	6261.37(0.00)(↑)
LWIQ+SPT	10156.43(0.00)(↑)	6274.87(0.00)(↑)	10104.80(0.00)(↑)	6250.91(0.00)(↑)
LWIQ+EDD	10142.49(0.00)(↑)	4272.82(0.00)(↑)	8779.17(0.00)(↑)	3587.16(0.00)(↑)
LWIQ+LWR	10680.63(0.00)(↑)	6479.56(0.00)(↑)	9455.17(0.00)(↑)	5539.43(0.00)(↑)
LWIQ+FIFO	12520.61(0.00)(↑)	7477.10(0.00)(↑)	11321.88(0.00)(↑)	6575.32(0.00)(↑)
DRL	2695.19(199.71)(↑)	863.03(223.10)(↑)	6249.33(271.75)(↑)	1970.43(352.45)(↑)
NichGPDRL	2446.60(149.44)	462.48(34.11)	5976.72(244.07)	1613.10(89.89)

minal within this rule, having been utilised four times. Following SLACK, both PT and WKR are employed twice, while NIQ is employed only once. The simplified representation of the sequencing rule 1 is denoted as S_1 and is expressed in Eq. (5.2). Given that NIQ, representing the number of operations in the waiting queue for a machine, remains constant for all candidate operations, it can be disregarded. This rule prioritises jobs that have a shorter processing time for the current operation, a smaller slack, and a higher percentage of the current processing time relative to the remaining processing time of the job.

$$S_1 = PT + NIQ + SLACK + \frac{SLACK}{1 + \frac{PT}{WKR}} + 1 \quad (5.2)$$

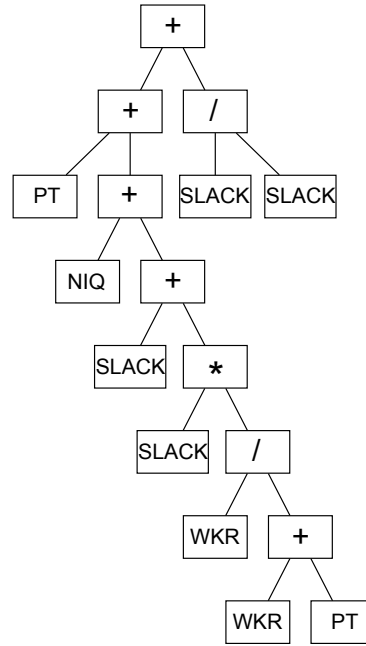


Figure 5.6: The sequencing rule 1 tree structure of four sequencing rules evolved by a NichGP run.

Regarding the sequencing rule 2 in Figure 5.7, it comprises five terminals: PT, SLACK, NPT, WKR, and TIS. Notably, SLACK emerges as the most frequently used terminal in this rule, having been employed four times. Subsequently, PT is used twice, while NPT, WKR, and TIS are each used only once. The simplified representation of the sequencing rule 2 is denoted as S_2 and is expressed in Eq. (5.3). The interpretation of this rule varies depending on specific scenarios. When the SLACK time of a job surpasses the sum of the remaining processing time and the time the job has spent in the system, the rule gives preference to jobs with shorter processing times, shorter slack time, and smaller processing times for their subsequent operations. Conversely, if the SLACK time of a job is less than the sum of its remaining processing time and the time it has stayed in the system, the rule favors jobs with shorter processing times, shorter slack time, smaller processing times for their next operations, shorter remaining

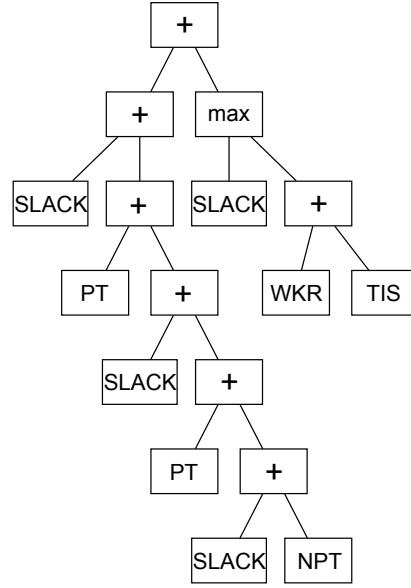


Figure 5.7: The sequencing rule 2 tree structure of four sequencing rules evolved by a NichGP run.

processing time, and shorter time spent in the system.

$$S_2 = 2 \times PT + 3 \times SLACK + NPT + \max\{SLACK, WKR + TIS\} \quad (5.3)$$

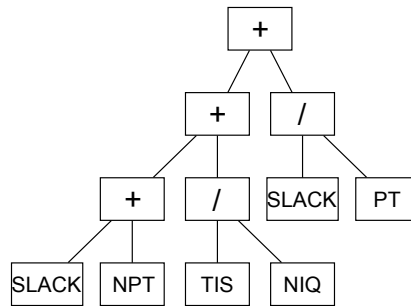


Figure 5.8: The sequencing rule 3 tree structure of four sequencing rules evolved by a NichGP run.

Concerning the sequencing rule 3 in Figure 5.8, it comprises five terminals (PT, SLACK, NPT, NIQ, and TIS), with SLACK being the predominant terminal, utilised twice. PT, NPT, NIQ, and TIS are each used only once. The simplified representation of the sequencing rule 3 is denoted as S_3 and is illustrated in Eq. (5.4). Since NIQ represents the number of operations in the waiting queue of the machine and remains constant for all candidate operations, it can be disregarded. This rule prioritises jobs with shorter slack time, smaller processing time for their next operation, shorter time spent in the system, and longer processing time.

$$S_3 = SLACK + NPT + \frac{TIS}{NIQ} + \frac{SLACK}{PT} \quad (5.4)$$

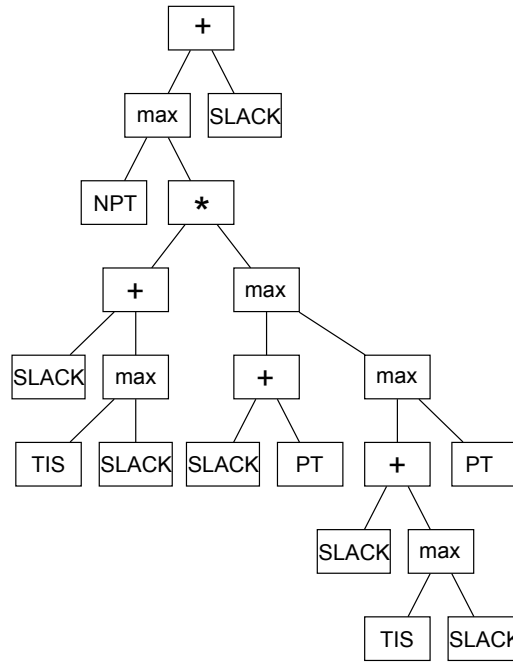


Figure 5.9: The sequencing rule 4 tree structure of four sequencing rules evolved by a NichGP run.

Regarding the sequencing rule 4 in Figure 5.9, it consists of four terminals (PT, SLACK, NPT, and TIS), with SLACK being the most frequently

utilised terminal, employed six times. PT follows with two occurrences, while NPT and TIS are each used only once. The simplified representation of the sequencing rule 4 is denoted as S_4 and is expressed in Eq. (5.5). This rule has different meanings in distinct situations. When the processing time of the next operation significantly exceeds the slack time, the current processing time, and the duration the job has spent in the system, it favors jobs with shorter processing times for the next operation and shorter slack times. Alternatively, under different circumstances, the rule prioritises jobs with shorter processing times, shorter slack times, and shorter durations spent in the system.

$$\begin{aligned}
 S_4 = & \max\{NPT, (SLACK + \max\{TIS, SLACK\}) \\
 & \times \max\{SLACK + PT, \max\{SLACK, \\
 & \max\{TIS, SLACK\}, PT\}\} + SLACK
 \end{aligned} \tag{5.5}$$

Based on the above analyses of various sequencing rules, it is evident that both slack time and current processing time are crucial features that hold significant importance in their decision-making criteria. Additionally, the processing time of the next operation and the time spent in the system are also important factors in selecting a candidate job for processing as the next task. Despite these similarities, there are notable differences. For instance, the sequencing rules 1 and 2 consider the remaining processing time of the job, while sequencing rules 3 and 4 do not take this into account. Furthermore, the sequencing rule 3 exhibits a preference for jobs with longer processing times, while all other rules favor jobs with shorter processing times. This observation indicates that these rules focus on distinct situations during the long-term scheduling process and exhibit different behaviours. The variations among these rules empower the proposed NichGPDL algorithm to make informed decisions in selecting an expert rule at specific decision points, thereby facilitating intelligent scheduling.

5.6 Chapter Summary

This chapter takes advantage of both GP and RL and presents a hybrid GP and RL method (NichGPDRL) to learn effective sequencing and routing agents for making intelligent selections among heuristics across different decision points to address the DFJSS problem. Specifically, the niching GP method is employed to learn a diverse set of high-quality scheduling heuristics. Subsequently, the sequencing rules derived from these learned scheduling heuristics serve as actions for the DRL method. This method tackles the challenge of RL that requires adapting to altered operations at various decision points, enabling the learning of intelligent agents capable of making optimal selections among these actions to generate effective schedules. It is important to note that this chapter serves as an initial exploration into integrating GP and RL. We extend an existing published DRL method (the baseline DRL) by incorporating the proposed NichGP, resulting in NichGPDRL. The baseline DRL learns an end-to-end routing agent to manage a fixed number of machines and a heuristic selection sequencing agent to handle the dynamically changing number of operations. In this case, our enhanced method focuses solely on replacing the manually designed sequencing rules with NichGP-evolved ones.

Comparative results against baseline DRL, end-to-end routing agent with GP evolved sequencing rules, and widely used manually designed scheduling heuristics validate the effectiveness of the proposed hybrid method. Additionally, comparisons against traditional GP-assisted DRL confirm the effectiveness of the proposed niching GP method. Furthermore, contrasting results against the proposed method without the DRL training process, where the sequencing rule is fixed as one of the candidate actions, verify the efficacy of the DRL learning process. Further analysis of action contribution demonstrates that the scheduling heuristics learned by the niching GP method contribute similar percentages to the overall performance. The behavioural analysis reveals that the proposed nich-

ing GP method can learn diverse scheduling heuristics compared to traditional GP methods. Structural analysis indicates that the learned scheduling heuristics by the niching GP exhibit similarities but also show distinct performances at different decision points. Overall, this chapter shows that the hybrid GP and RL method contributes to the development of effective agents for solving the DFJSS problem.

With the assistance of GP, the capability of RL for DFJSS has been significantly enhanced. This, in turn, extends GP's capability to contribute positively to DRL for DFJSS. However, the performance of RL + GP still cannot match GP's performance for these DFJSS problems. Future work will be necessary to further investigate more effective combinations of GP and RL to achieve better performance than both GP and RL.

Chapter 3 introduces three diversity-based parent selection mechanisms for GP to produce high-quality offspring and ultimately evolve effective single scheduling heuristics for addressing the DFJSS problems. In Chapter 4, an ensemble GP method is developed to evolve a group of scheduling heuristics for making joint decisions to solve the DFJSS problems. This chapter further explores the utilisation of a group of scheduling heuristics evolved by GP, employing RL to select appropriate scheduling heuristics among the diverse ones at different decision points. All these chapters solely focus on solving single-objective DFJSS problems. In the next chapter, we propose two novel multi-objective GP algorithms designed to effectively tackle multi-objective DFJSS problems.

Chapter 6

Multi-objective Genetic Programming

This chapter proposes new methods using GP and multi-objective solving techniques for solving the multi-objective DFJSS (MO-DFJSS) problems.

6.1 Introduction

In practical manufacturing, for some situations, DFJSS needs to consider different objectives in parallel [261], which is known as multi-objective DFJSS (MO-DFJSS). In MO-DFJSS, objectives are usually conflicted, such as mean-flowtime (F_{mean}) and max-tardiness (T_{max}), which is a common difficulty in multi-objective optimisation. The study of MO-DFJSS is of great practical importance. However, the study on MO-DFJSS is limited.

Currently, there are only a few studies about incorporating well-known Pareto dominance-based methods (i.e., non-dominated sorting genetic algorithm II [52] and strength Pareto evolutionary algorithm 2 [289]) into GP, named NSGPII [267] and SPGP2 [267], for MO-DFJSS. They can successfully evolve a Pareto front of scheduling heuristics for solving the MO-DFJSS problems. However, Pareto-dominance algorithms can sometimes have difficulty in preserving the spreadability/diversity of the Pareto

front, which is crucial for generating a representative set of solutions [288, 127]. Real-world decision-making often involves revising priorities or weights assigned to different objectives. A well-spread set of solutions provides more flexibility in such situations. If solutions are clustered together, changes in preference might leave users with very few options [106]. This limitation of Pareto-dominance algorithms suggests the need for a method that can provide great spreadability of the evolved Pareto front of scheduling heuristics.

Multi-objective evolutionary algorithm based on decomposition (MOEA/D) is a well-known multi-objective method [276]. The key idea of MOEA/D is to decompose a multi-objective optimisation problem into a number of sub-problems via a scalar function and then optimise them simultaneously [85]. The strength of MOEA/D that makes it have the potential to deal with the difficulties that Pareto dominance-based algorithms cannot handle. MOEA/D uses a decomposition approach that allows the optimisation of multiple sub-problems in different directions simultaneously, making it possible to generate a spread Pareto front. Further, the high search capability of MOEA/D, especially for difficult multi-objective problems, has been repeatedly reported [104, 105, 126]. Some works have shown the potential of using MOEA/D algorithms for solving static scheduling problems [37, 108, 235, 279]. However, to the best of our knowledge, there is no study using MOEA/D for solving the MO-DFJSS problem. It is interesting to investigate the incorporation of MOEA/D and GP for solving the DFJSS problem.

In addition, among the existing multi-objective algorithms for DFJSS, the NSGP-II showed the best performance in terms of HV [290] and IGD [134]. However, NSGP-II acts on the genotype of individuals and does not consider semantic information, which reflects the behaviour of the genotype. Semantic GP [217] has been proposed to enhance population diversity by integrating semantic information into the evolutionary process. Its effectiveness has been demonstrated across diverse domains, includ-

ing symbolic regression [214], classification [15, 192], and feature selection [169]. However, to the best of our knowledge, semantic information has not been incorporated into NSGP-II for solving the MO-DFJSS problem. Given these promising results, it becomes particularly intriguing to explore how to improve the performance of NSGP-II for MO-DFJSS by incorporating semantic information.

6.1.1 Chapter Goals

The goal of this chapter is to *propose two novel multi-objective GP methods for solving the MO-DFJSS with multiple objectives*. The proposed two multi-objective GP methods are expected to evolve a Pareto front of effective scheduling heuristics. To be specific, the objectives of this chapter are as follows.

1. Propose two novel multi-objective GP methods (i.e., MOGP/D and semantic NSGP-II) for evolving a Pareto front of effective scheduling heuristics for MO-DFJSS. The MOGP/D is developed by adapting ideas from the classical MOEA/D [276] along with the characteristics of MO-DFJSS. The semantic NSGP-II is built upon the original NSGP-II.
2. Design a mapping strategy to match individuals to sub-problems (MO-DFJSS considering different weight combinations) for the MOGP/D for reducing the effects of seed rotation in MO-DFJSS.
3. Define the *semantic* and *semantic distance* concepts in the context of the MO-DFJSS domain for the proposed semantic NSGP-II, design strategies to measure the semantic information derived from MO-DFJSS, and design strategies to use the semantic information during the evolution process.
4. Verify the effectiveness of the proposed MOGP/D and semantic NS-

GPII methods by comparing them with the state-of-the-art Pareto-dominance multi-objective method.

5. Analyse how semantic information affects the performance of evolved scheduling heuristics by NSGPPII on solving the MO-DFJSS problem.

6.1.2 Chapter Organisation

The rest of this chapter is organised as follows. Detailed description of the proposed two algorithms are given in Section 6.2. The experimental design is provided in Section 6.3, followed by results and discussions in Section 6.4. Further analyses are conducted in Section 6.5. Finally, Section 6.6 concludes this chapter.

6.2 Proposed Algorithms

6.2.1 MOGP/D

In this section, we hybrid MOEA/D with GP with multi-tree representation to evolve effective Pareto front of scheduling heuristics on solving the MO-DFJSS problem. The proposed algorithm is named MOGP/D. The flowchart of the proposed MOGP/D can be seen in Figure 6.1.

Following the main strategies of MOEA/D, MOGP/D has population initialisation, sub-problem decomposition, fitness evaluation, fitness normalisation, parent selection, and breeding. Different from MOEA/D, combined with strategies and features of GP, also considering the characteristic of MO-DFJSS, MOGP/D has the training process and the test process, searches in the heuristic space, ignores the neighbor replacement strategy, develops a mapping strategy, and uses the tournament selection.

To be specific, the training process finally output Pareto front of scheduling heuristics evolved on the training instances. Then the Pareto

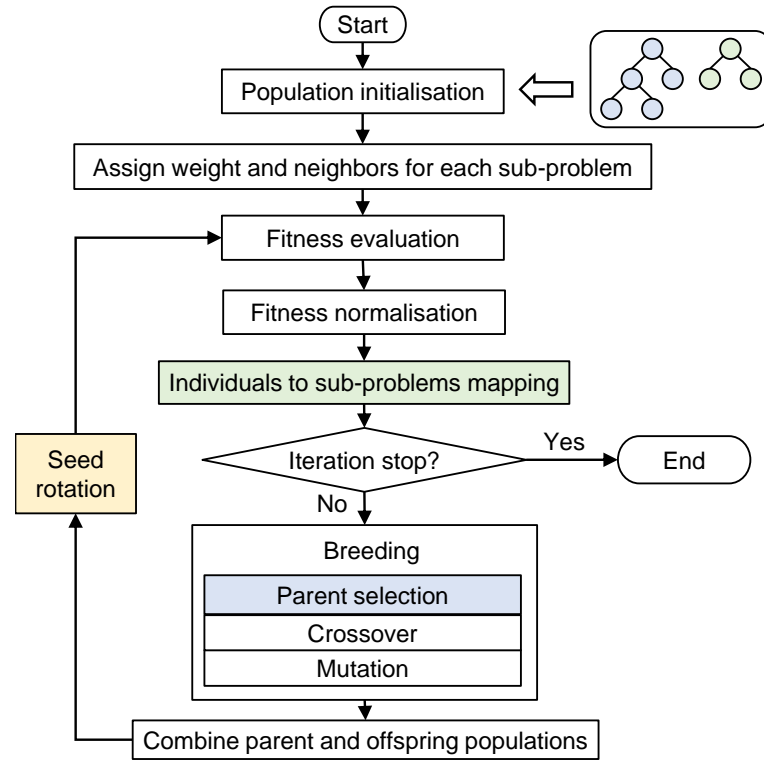


Figure 6.1: The flowchart of MOGP/D.

front of scheduling heuristics are used on unseen test instances to measure their performance. In MOGP/D, each individual has two trees, representing the routing rule and sequencing rule respectively, which are combined by features in the heuristic space. In addition, MOGP/D changes the training instance (rotates seed) for each new generation which is a normal strategy in the domain of GP for DFJSS, which helps improve the generalisation ability of the learned scheduling heuristics. However, this strategy might destroy the relationship among neighbors, as the fit scheduling heuristic for each sub-problem in the current generation might not fit the sub-problem in the next generation anymore. To avoid this situation, a mapping strategy is proposed to match individuals to sub-problems after fitness evaluation. To generate offspring, tournament selection is used to select good parents for breeding. Traditional MOEA/D evaluates off-

spring immediately after it is generated and uses the offspring to replace the parents/neighbors if it outperforms the parents/neighbors. Thus, the survival of the fittest proceeds within the breeding process of each generation, and no tournament selection is needed to choose parents. However, GP prefers to reserve all the offspring. Therefore, MOGP/D does not adopt the neighbor replacement strategy, instead, it reserves all the offspring. As we change the training instance for every new generation in MOGP/D, before proceeding with the survival process and mapping, we combine parents and offspring to do fitness evaluation. Fitness normalisation is conducted to reduce the influence of objective bias. Then, the mapping process will select the fittest individual for each sub-problem. After mapping, the population will shrink back to the original population size. Finally, the non-dominated individuals from the final generation will be output and used on the test set to measure their performance. The fitness normalisation and mapping strategies are described as follows in detail.

Fitness Normalisation

Since it is not easy to estimate the ideal points and nadir points of a real-world problem, especially, when we change the training instance for each new generation in MO-DFJSS. In this chapter, we use the *ratio* $f_{i,t}/b_{i,t}$ of the objective value of each individual $f_{i,t}$ versus the objective value obtained by the baseline rules $b_{i,t}$ as the normalised fitness $obj_{i,t}$ on the objective t . We select one routing rule for all the objectives, which is the Least work remaining in queue (LWIQ) rule, and five specific sequencing rules for the five objectives, which are listed in Table 6.1.

The Mapping Strategy

The mapping strategy is used to map individuals to sub-problems after moving to a new generation. The mapping process proceeds after fitness normalisation, the details can be seen in Algorithm 8.

Table 6.1: The baseline rules for fitness normalisation.

Objective	Sequencing rule
Fmax	FCFS: first come first serve
Fmean	SPT: shortest processing time
Tmax	EDD: earliest due date
Tmean	SlackperRPTplusPT: slack per remaining processing time plus processing time
WTmean	WATC: weighted apparent tardiness cost [218]

We can see that, the individuals \mathcal{P} from parents and offspring are combined together and divided into two groups \mathcal{A} and \mathcal{B} . The group \mathcal{A} contains individuals with normal objective values, while the group \mathcal{B} stores poor individuals with Inf. objective values (Line 1). We generate a random permutation Ω^* of sub-problems Ω (Line 2). For each sub-problem, we calculate the weight-sum objective value (Line 7) and select the individual with the smallest weight-sum objective from the group \mathcal{A} (Line 9) if the group \mathcal{A} is not empty. Otherwise, an individual in the group \mathcal{B} is randomly selected (Line: 14) and assigned to this sub-problem. The selected individual will then be deleted from its original group (Lines 11 and 15). The whole process goes on until each sub-problem is assigned an individual. Totally N individuals are selected for the N sub-problems and the remaining N individuals will be deleted.

6.2.2 Semantic NSGP II

This section introduces the proposed semantic NSGP II method. The proposed method uses the NSGP II parent selection, crossover, and mutation to generate offspring for the next generation. On top of that, it designs novel strategies to decide which kind of offspring is allowed to be added to the next generation by considering semantic diversity and semantic similarity. In this section, we begin by providing the definitions of *semantic* and *semantic distance* in MO-DFJSS, then describe the proposed strategies. The

Algorithm 8: Mapping individuals to sub-problems.

Input: The population \mathcal{P} combining parents and offspring with $2N$ individuals, Ω includes N sub-problems.

Output: Matched pairs $\{p_k, \alpha_j\}$ between each individual $p_k \in \mathcal{P}$ and each sub-problem $\alpha_j \in \Omega$.

- 1 Divide individuals in \mathcal{P} into two groups, one group \mathcal{A} contains a individuals with normal fitness, and the other group \mathcal{B} stores b individuals with Inf. fitness (bad run), where $a + b = 2N$;
- 2 Generate a random permutation Ω^* of sub-problems Ω ;
- 3 **foreach** sub-problem $\alpha_j \in \Omega^*$ **do**
- 4 Obtain the weight vector w_j for sub-problem α_j ;
- 5 **if** \mathcal{A} is nonempty **then**
- 6 **foreach** individual $p_i \in \mathcal{A}$ **do**
- 7 Calculate the weight-sum objective value η_i^j of the individual p_i on the sub-problem α_j : $\eta_i^j = \sum_{t=1}^m w_{j,t} * obj_{i,t}$, where $obj_{i,t}$ denotes the normalised fitness of individual p_i on the objective t ;
- 8 **end**
- 9 $k = \operatorname{argmin}_{i=1, \dots, a} (\eta_i^j)$;
- 10 Individual p_k is assigned to sub-problem α_j ;
- 11 $\mathcal{A} = \mathcal{A} / p_k$;
- 12 **end**
- 13 **else**
- 14 Random select an individual p_r from \mathcal{B} and assign it to sub-problem α_j ;
- 15 $\mathcal{B} = \mathcal{B} / p_r$;
- 16 **end**
- 17 **end**

flowchart of the improved NSGP-II with the proposed strategies is shown in Figure 6.2.

Semantic in MO-DFJSS

This chapter defines the *semantic* in MO-DFJSS as the phenotypic characterisation (PC) [260]. In the research domain of DFJSS, PC [260] serves as a common method to describe an individual's behaviour. Although the

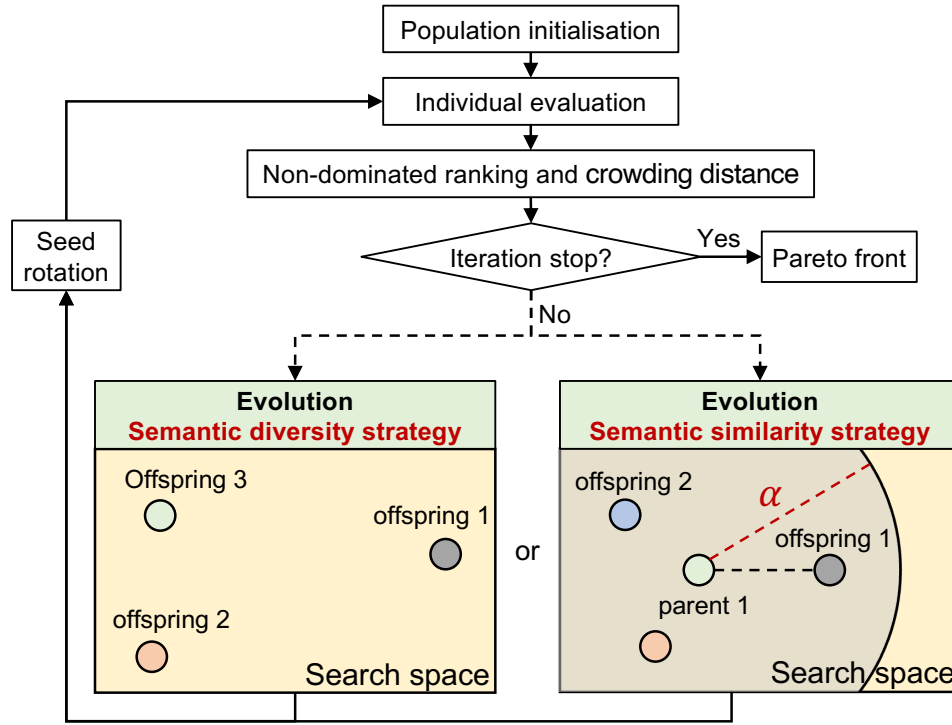


Figure 6.2: The flowchart of the proposed NSGPII with the semantic diversity strategy or the semantic similarity strategy for evolution.

PC is employed to analyse population diversity by the proposed methods in Chapter 3, it has not been thoroughly introduced. This chapter provides a comprehensive understanding of the PC. The PC represents a list of decisions given by an individual on a given number of decision points. These decision points are derived by applying a reference scheduling heuristic to a given DFJSS instance. Specifically, this chapter employs the weighted shortest processing time (WSPT) as the reference sequencing rule and working remaining in the queue (WIQ) as the reference routing rule. Considering that each instance often contains thousands of decision points, to save time and ease of use, we randomly select 20 sequencing decision points and 20 routing decision points, each involving a set of 7 candidates (operations for sequencing rule and machines for routing rule).

Table 6.2: An example of calculating the PC of an individual.

Sequencing				Routing			
Decision points	Reference rule	Sequencing rule	Decision	Decision points	Reference rule	Routing rule	Decision
1(O_1)	1	3	3	1(M_1)	<u>2</u>	<u>1</u>	2
1(O_2)	<u>3</u>	<u>1</u>		1(M_2)	1	2	
1(O_3)	2	2		1(M_3)	3	3	
2(O_1)	3	2	1	2(M_1)	2	2	1
2(O_2)	<u>1</u>	<u>1</u>		2(M_2)	3	3	
2(O_3)	2	3		2(M_3)	<u>1</u>	<u>1</u>	
3(O_1)	1	2	2	3(M_1)	1	2	3
3(O_2)	<u>2</u>	<u>1</u>		3(M_2)	<u>3</u>	<u>1</u>	
3(O_3)	3	3		3(M_3)	2	3	

Then, to calculate the PC of an individual, the sequencing/routing rule in an individual is applied to these decision points, and the ranks of the selected operations/machines across these decision points are utilised to construct the PC. An illustrative example of calculating the PC for an individual is shown in Table 6.2, considering 3 sequencing decision points and 3 routing decision points based on the reference scheduling heuristic. According to the given description, the PC of this example is a combination of sequencing decisions and routing decisions, which is [3, 1, 2, 2, 1, 3].

Based on the PC, the *semantic distance* between individuals ind_a and ind_b is defined as the number of different decisions between their semantics and can be calculated based on Eq. (6.1). In contrast to other semantic methods that typically rely on Euclidean distance for calculating semantic differences, this chapter proposes this definition because the Euclidean distance between machine or operation rankings in semantics is considered meaningless in DFJSS.

$$dis_{a,b} = \sum_{i=1}^{40} d_{a,b,i} \quad \text{where} \quad d_{a,b,i} = \begin{cases} 0 & \text{if } pc_{a,i} = pc_{b,i} \\ 1 & \text{otherwise} \end{cases} \quad (6.1)$$

Semantic Diversity Strategy

This strategy aims to enhance the semantic diversity of the population by only accepting offspring that are semantically different from each other. Two individuals are considered semantically different if their semantic distance is greater than 0. This strategy is used whenever an offspring is generated by crossover or mutation. To be specific, when an offspring ind_o is generated, we compare it with all the other offspring already generated at the current generation. If ind_o is found to be semantically different from all the existing ones, it is accepted as a new offspring. However, if any duplicates are found, indicating that it is not semantically different, the offspring ind_o is discarded. This process is repeated iteratively until the population is filled with semantically different offspring.

Semantic Similarity Strategy

This strategy builds upon the aforementioned semantic diversity approach and introduces an additional constraint to control the *semantic similarity* between generated offspring and their parents. The degree of semantic similarity between individuals is restricted by a threshold α . Specifically, when an offspring ind_o is generated, we first compare it with all the previously generated offspring in the current generation. If ind_o is found to be semantically different from all existing ones, we further examine whether it is semantically similar to at least one of its parents. This is done by assessing whether the semantic distance between ind_o and its parent ind_p is smaller than the threshold α . If ind_o is similar to any of its parents, then it is accepted as a new offspring; otherwise, it is discarded. This process is iterated until the offspring population is filled. The idea behind this strategy is that by limiting the similarity between offspring with their parents, we expect evolution to be smooth, without losing convergence, and at the same time maintain diversity. To achieve this goal, the key point is to determine an appropriate α value.

6.3 Experimental Design

The dataset employed for simulating various job shop scheduling scenarios, along with the terminals and functions utilised for constructing individuals in this chapter, remains consistent with Chapter 3. Regarding the parameter configurations, the population size is set as 1000, and the Pareto front is output after 50 generations. The Ramped-half-and-half method is employed for population initialisation. The crossover and mutation rates are set to 0.85 and 0.15, respectively. Parent selection is performed using tournament selection with a size of 7. In addition, the experimental design of this chapter is outlined as follows.

6.3.1 Performance Measures

The three key indicators used for verifying the proposed multi-objective algorithms are *Hypervolume*, *Inverted Generational Distance*, and *Maximum Spread*. Their principles are described below.

1. **Hyper Volume (HV) [290]:** Given a solution set A and a set of reference points Z^* , HV describes the size of the space covered:

$$HV(A) = \lambda \left(\bigcup_{a \in A} x | a \prec x \prec z \right) \quad (6.2)$$

where λ indicates the Lebesgue measure. $a \prec x \prec z$ denotes x dominates z and is dominated by a . A high HV value is preferable.

2. **Inverted Generational Distance (IGD) [134]:** Given a solution set A and a set of reference points Z^* , IGD measures the distance from the reference points to the solution set:

$$IGD(A, Z^*) = \frac{\sum_{i=1}^m \min_{a \in A} d(z_i^*, a)}{m} \quad (6.3)$$

where $d(z_i, a)$ represents the Euclidean distance between z_i and a . We prefer the algorithm with a low IGD value.

3. **Maximum Spread (MS)** [127]: Let A be a solution set to a multi-objective problem, MS measures the range of A by considering the maximum extent of each objective:

$$MS(A) = \sqrt{\sum_{j=1}^m \max_{a, a' \in A} (a_j - a'_j)^2} \quad (6.4)$$

The higher the MS, the better the algorithm.

6.3.2 Comparison Design

This chapter proposes a MOGP/D algorithm and proposes two strategies to incorporate semantic information into NSGP-II for MO-DFJSS, which are the semantic diversity strategy and the semantic similarity strategy. To simplify the algorithm description, we name NSGP-II with semantic diversity strategy as NSGP-II^d, and NSGP-II with semantic similarity strategy as NSGP-II^s. To verify the effectiveness of MOGP/D, NSGP-II^d, and NSGP-II^s, they are compared with the original (state-of-the-art) NSGP-II algorithm. Besides, for NSGP-II^s, different α values are tested, including 2, 4, 6, 8, 10, 12, and 14.

To measure and compare the performance of algorithms, we conducted 30 independent runs for each algorithm and employed Friedman's test and Wilcoxon rank-sum test for comparison. If Friedman's test yielded significant results, we proceed with the Wilcoxon rank-sum test for pairwise comparisons between the improved NSGP-II considering the semantic diversity strategy or the semantic similarity strategy and the classical NSGP-II, using a significance level of 0.05. In the subsequent results, we use the symbols " \uparrow ", " \downarrow ", and " $=$ " to indicate statistical significance, denoting better, worse, or similar results compared to their counterparts, respectively.

In this chapter, six scenarios are examined by considering different combinations of objectives and different utilisation levels (0.85 and 0.95),

which can refer to Table 6.3. For each scenario, 50 instances are used for training, while a separate set of 50 unseen instances is used for the test.

Table 6.3: Six scenarios.

Scenario	Objective 1	Objective 2	Utilisation
1	max-flowtime	max-weighted-tardiness	0.85
2	max-flowtime	max-weighted-tardiness	0.95
3	max-weighted-flowtime	max-tardiness	0.85
4	max-weighted-flowtime	max-tardiness	0.95
5	mean-flowtime	mean-weighted-tardiness	0.85
6	mean-flowtime	mean-weighted-tardiness	0.95

6.4 Results and Discussions

6.4.1 Test Performance

Tables 6.4, 6.5, and 6.6 present the mean and standard deviation of the HV, IGD, and MS results for different algorithms across 30 independent runs on the test instances of the six scenarios. The bottom of the tables shows the results of the Wilcoxon comparison and Friedman's test. For NSGP^{II}^s, we expect evolution to be smooth, without losing convergence, and at the same time maintain diversity. To achieve this goal, the key point is to determine an appropriate α value. Therefore, we first analyse the effect of α on NSGP^{II}^s. Then we compare NSGP^{II}, MOGPD, NSGP^{II}^d, and NSGP^{II}^s with the best α .

It is evident from the HV results presented in Table 6.4 that NSGP^{II}^s with $\alpha = 6$ exhibits markedly superior performance in terms of HV across five scenarios compared to NSGP^{II} and obtains statistically similar HV performance on the remaining one scenario. NSGP^{II}^s with $\alpha = 8$, $\alpha = 10$, $\alpha = 12$, and $\alpha = 14$ demonstrate significantly better HV performance than NSGP^{II} on four scenarios, with statistically similar HV performance ob-

Table 6.4: The mean (standard deviation) test HV of 30 independent runs of NSGP_{II}, MOGP/D, NSGP_{II}^d and NSGP_{II}^s with different α for 6 scenarios.

Scenario	NSGP _{II}	MOGP/D	NSGP _{II} ^d	NSGP _{II} ^s	
				$\alpha = 2$	$\alpha = 4$
1	0.84(0.04)	0.81(0.06)	0.86(0.04)	0.76(0.07)	0.84(0.03)
2	0.82(0.03)	0.78(0.05)	0.85(0.03)	0.74(0.06)	0.82(0.03)
3	0.87(0.03)	0.88(0.03)	0.88(0.03)	0.82(0.04)	0.86(0.03)
4	0.95(0.01)	0.95(0.01)	0.95(0.01)	0.93(0.02)	0.95(0.01)
5	0.59(0.23)	0.58(0.23)	0.58(0.18)	0.57(0.17)	0.62(0.12)
6	0.98(0.01)	0.98(0.02)	0.98(0.01)	0.97(0.01)	0.98(0.01)
↑/= / ↓	-	0/4/2	2/4/0	0/1/5	0/6/0
rank	7	7.58	6.17	10	6.92

Scenario	NSGP _{II} ^s				
	$\alpha = 6$	$\alpha = 8$	$\alpha = 10$	$\alpha = 12$	$\alpha = 14$
1	0.86(0.03)	0.86(0.04)	0.86(0.04)	0.87(0.03)	0.86(0.04)
2	0.86(0.03)	0.86(0.03)	0.86(0.03)	0.86(0.02)	0.85(0.03)
3	0.89(0.04)	0.89(0.03)	0.89(0.03)	0.89(0.03)	0.89(0.02)
4	0.96(0.01)	0.96(0.02)	0.96(0.01)	0.96(0.01)	0.96(0.01)
5	0.73(0.09)	0.66(0.12)	0.70(0.14)	0.59(0.18)	0.69(0.14)
6	0.98(0.01)	0.98(0.01)	0.98(0.01)	0.98(0.01)	0.98(0.01)
↑/= / ↓	5/1/0	4/2/0	4/2/0	4/2/0	4/2/0
rank	3.08	3.58	3.25	3.5	3.92

served on the remaining two scenarios. NSGP_{II}^s with $\alpha = 4$ obtains statistically similar HV performance as NSGP_{II} on all six scenarios. However, NSGP_{II}^s with $\alpha = 2$ obtains statistically similar HV performance as NSGP_{II} on only one scenario and performs significantly worse than NSGP_{II} on the remaining five scenarios. According to the Friedman's test results, NSGP_{II}^s with $\alpha = 6$ achieves the highest rank, followed by NSGP_{II}^s with $\alpha = 10$, NSGP_{II}^s with $\alpha = 12$, NSGP_{II}^s with $\alpha = 8$, NSGP_{II}^s with $\alpha = 14$, NSGP_{II}^s with $\alpha = 4$, and finally NSGP_{II}^s with $\alpha = 2$. Given that NSGP_{II}^s with $\alpha = 6$ demonstrates the best performance among all NSGP_{II}^s variants with different α values, further comparisons are made

with other algorithms. Comparing NSGP^{II}^s with $\alpha = 6$ to other algorithms, it is evident that MOGP/D yields significantly inferior HV performance than NSGP^{II} on two scenarios and shows statistically similar HV performance on the remaining five scenarios. NSGP^{II}^d exhibits significantly better HV performance than NSGP^{II} on two scenarios, with similar HV performance observed on the remaining four scenarios. Notably, NSGP^{II}^s with $\alpha = 6$ surpasses both of these methods in terms of HV. Furthermore, NSGP^{II}^s consistently demonstrates significantly better HV performance compared to NSGP^{II} across five scenarios. As per the Friedman's test results, NSGP^{II}^s with $\alpha = 6$ ranks highest among these four methods, followed by NSGP^{II}^d, NSGP^{II}, and MOGP/D.

In terms of IGD performance (refer to Table 6.5), NSGP^{II}^s with $\alpha = 6$, $\alpha = 8$, $\alpha = 10$, $\alpha = 12$, $\alpha = 14$ demonstrate significantly superior performance compared to NSGP^{II} across four scenarios and obtains statistically similar IGD performance on the remaining two scenarios. NSGP^{II}^s with $\alpha = 4$ exhibit significantly better IGD performance than NSGP^{II} on one scenario, while displaying statistically similar IGD performance on the remaining five scenarios. In contrast, NSGP^{II}^s with $\alpha = 2$ performs worse than NSGP^{II} on five scenarios and shows statistically comparable IGD performance on one scenario. According to the Friedman's test results, NSGP^{II}^s with $\alpha = 8$ achieves the highest rank, followed by NSGP^{II}^s with $\alpha = 6$, NSGP^{II}^s with $\alpha = 10$, NSGP^{II}^s with $\alpha = 14$, NSGP^{II}^s with $\alpha = 12$, NSGP^{II}^s with $\alpha = 4$, and finally NSGP^{II}^s with $\alpha = 2$. As NSGP^{II}^s with $\alpha = 8$ demonstrates the best performance among all NSGP^{II}^s variants with different α values, further comparisons are conducted with other algorithms. When compared with other algorithms, it is evident that MOGP/D yields significantly worse IGD performance than NSGP^{II} on half scenarios and shows statistically similar IGD performance on the remaining half scenarios. NSGP^{II}^d demonstrates significantly better IGD performance than NSGP^{II} on two scenarios, while achieving similar IGD performance to NSGP^{II} on the remaining four scenarios. No-

Table 6.5: The mean (standard deviation) test IGD of 30 independent runs of NSGP_{II}, MOGP/D, NSGP_{II}^d and NSGP_{II}^s with different α for 6 scenarios.

Scenario	NSGP _{II}	MOGP/D	NSGP _{II} ^d	NSGP _{II} ^s	
				$\alpha = 2$	$\alpha = 4$
1	0.11(0.03)	0.13(0.05)	0.10(0.03)	0.14(0.04)	0.09(0.02)
2	0.11(0.02)	0.15(0.04)	0.09(0.02)	0.16(0.04)	0.11(0.02)
3	0.07(0.02)	0.08(0.03)	0.07(0.02)	0.10(0.03)	0.07(0.02)
4	0.03(0.01)	0.04(0.01)	0.03(0.01)	0.05(0.02)	0.03(0.01)
5	0.30(0.23)	0.31(0.23)	0.29(0.17)	0.30(0.17)	0.24(0.10)
6	0.01(0.01)	0.01(0.01)	0.01(0.01)	0.02(0.01)	0.01(0.00)
↑/= / ↓	-	0/3/3	2/4/0	0/1/5	1/5/0
rank	7.08	8.5	5.67	9.75	5.58
Scenario	NSGP _{II} ^s				
	$\alpha = 6$	$\alpha = 8$	$\alpha = 10$	$\alpha = 12$	$\alpha = 14$
1	0.10(0.03)	0.09(0.03)	0.10(0.02)	0.09(0.02)	0.09(0.03)
2	0.09(0.02)	0.09(0.02)	0.09(0.02)	0.09(0.01)	0.10(0.02)
3	0.06(0.02)	0.06(0.02)	0.06(0.01)	0.06(0.02)	0.06(0.01)
4	0.02(0.01)	0.02(0.01)	0.02(0.00)	0.03(0.01)	0.02(0.01)
5	0.16(0.07)	0.22(0.10)	0.19(0.11)	0.29(0.18)	0.20(0.13)
6	0.01(0.00)	0.01(0.01)	0.01(0.01)	0.01(0.01)	0.01(0.01)
↑/= / ↓	4/2/0	4/2/0	4/2/0	4/2/0	4/2/0
rank	3.42	3.33	3.58	4.42	3.67

tably, NSGP_{II}^s with $\alpha = 8$ exhibits even better performance. Furthermore, NSGP_{II}^s with $\alpha = 8$ consistently demonstrates significantly better IGD performance compared to NSGP_{II} across four scenarios. Based on the Friedman's test results, NSGP_{II}^s with $\alpha = 8$ ranks first among these four methods, followed by NSGP_{II}^d, NSGP_{II}, and MOGP/D.

In terms of MS performance (refer to Table 6.6), among all the variants of NSGP_{II}^s with different α , NSGP_{II}^s with $\alpha = 2$ and $\alpha = 4$ exhibit significantly worse performance compared to NSGP_{II} on two scenarios, while showing statistically similar MS on four scenarios. NSGP_{II}^s with $\alpha = 6$, $\alpha = 8$, $\alpha = 10$, and $\alpha = 12$ demonstrate significantly inferior

Table 6.6: The mean (standard deviation) test MS of 30 independent runs of NSGP_{II}, MOGP/D, NSGP_{II}^d and NSGP_{II}^s with different α for 6 scenarios.

Scenario	NSGP _{II}	MOGP/D	NSGP _{II} ^d	NSGP _{II} ^s	
				$\alpha = 2$	$\alpha = 4$
1	0.45(0.18)	0.59(0.19)	0.38(0.14)	0.37(0.12)	0.49(0.16)
2	0.63(0.16)	0.75(0.13)	0.60(0.15)	0.53(0.18)	0.58(0.14)
3	0.22(0.11)	0.23(0.06)	0.19(0.07)	0.17(0.11)	0.16(0.07)
4	0.17(0.07)	0.23(0.12)	0.16(0.09)	0.16(0.11)	0.17(0.08)
5	0.11(0.08)	0.16(0.07)	0.09(0.07)	0.08(0.08)	0.08(0.06)
6	0.01(0.01)	0.01(0.01)	0.01(0.01)	0.01(0.01)	0.01(0.01)
$\uparrow/=/\downarrow$	-	4/2/0	0/6/0	0/4/2	0/4/2
rank	3.58	1.75	5.92	7.33	5.75
Scenario	NSGP _{II} ^s				
	$\alpha = 6$	$\alpha = 8$	$\alpha = 10$	$\alpha = 12$	$\alpha = 14$
1	0.41(0.17)	0.44(0.18)	0.46(0.17)	0.42(0.18)	0.37(0.17)
2	0.65(0.13)	0.57(0.13)	0.65(0.18)	0.61(0.17)	0.61(0.14)
3	0.16(0.05)	0.19(0.10)	0.15(0.08)	0.16(0.05)	0.16(0.07)
4	0.15(0.06)	0.13(0.06)	0.17(0.10)	0.14(0.06)	0.13(0.05)
5	0.10(0.07)	0.10(0.08)	0.08(0.07)	0.12(0.09)	0.08(0.09)
6	0.01(0.01)	0.01(0.01)	0.01(0.01)	0.01(0.01)	0.01(0.01)
$\uparrow/=/\downarrow$	0/5/1	0/5/1	0/5/1	0/5/1	0/2/4
rank	5.67	6.17	5.42	5.75	7.67

MS performance than NSGP_{II} on one scenario, while displaying statistically similar MS performance on the remaining five scenarios. Moreover, NSGP_{II}^s with $\alpha = 14$ obtains significantly worse performance compared to NSGP_{II} on four scenarios, while showing statistically similar MS on the remaining two scenarios. According to the Friedman's test results, NSGP_{II}^s with $\alpha = 10$ achieves the highest rank among all the variants of NSGP_{II}^s with different α , followed by NSGP_{II}^s with $\alpha = 6$, $\alpha = 4$ and $\alpha = 14$ which rank equally, NSGP_{II}^s with $\alpha = 8$, NSGP_{II}^s with $\alpha = 2$, and NSGP_{II}^s with $\alpha = 14$. As NSGP_{II}^s with $\alpha = 10$ demonstrates the best MS performance among all NSGP_{II}^s variants with different α values, further

comparisons are conducted with other algorithms. When compared with other algorithms, it is evident that MOGP/D yields significantly better MS performance than NSGPPII on four scenarios and shows statistically similar MS performance on the remaining two scenarios. NSGPPII^d exhibits statistically similar MS performance to NSGPPII across all six scenarios. NSGPPII^s with $\alpha = 10$ shows slightly inferior performance. It demonstrates significantly worse MS performance compared to NSGPPII on one scenario and shows statistically similar MS performance on the remaining five scenarios. Based on the Friedman's test results, MOGP/D ranks first among these four methods, followed by NSGPPII, NSGPPII^s with $\alpha = 10$, and NSGPPII^d.

In summary, MOGP/D exhibits a broader exploration of the Pareto front of scheduling heuristics (MS) compared to NSGPPII^d, NSGPPII^s, and NSGPPII. Concerning the covered space (HV), NSGPPII^s demonstrates the highest performance, significantly outperforming MOGP/D, NSGPPII^d, and NSGPPII. NSGPPII^d shows significantly better performance than NSGPPII and MOGP/D, with MOGP/D exhibiting the least favorable results. When considering the distance to the reference points (IGD), NSGPPII^s consistently achieves superior performance, significantly surpassing MOGP/D, NSGPPII^d, and NSGPPII. NSGPPII^d also outperforms NSGPPII and MOGP/D in this regard, with MOGP/D displaying the least favorable outcomes. Overall, when considering the HV metric, NSGPPII^s with $\alpha = 6$ attains the highest performance, followed by NSGPPII^d and NSGPPII. MOGP/D exhibits the least favorable performance. For convenience, throughout the subsequent analysis, we use NSGPPII^s to denote NSGPPII^s with $\alpha = 6$.

Through the above analysis, we can see that semantic information plays an important role in improving the HV and IGD performance of NSGPPII on the MO-DFJSS problem. The HV and IGD performance of NSGPPII can be improved by increasing the diversity of the behaviours of the individuals in the population. Moreover, requiring the offspring to have

similar semantic behaviour with their parents can further improve the HV and IGD performance of NSGP^{II} in solving the MO-DFJSS problem. This finding highlights the positive impact of considering semantic diversity and semantic similarity in NSGP^{II} on addressing the MO-DFJSS problem. However, the incorporation of semantic information does not contribute to improving the MS performance. Despite MOGP/D exhibiting the worst HV and IGD performance, it attains the highest MS performance.

6.5 Further Analyses on Population Distribution

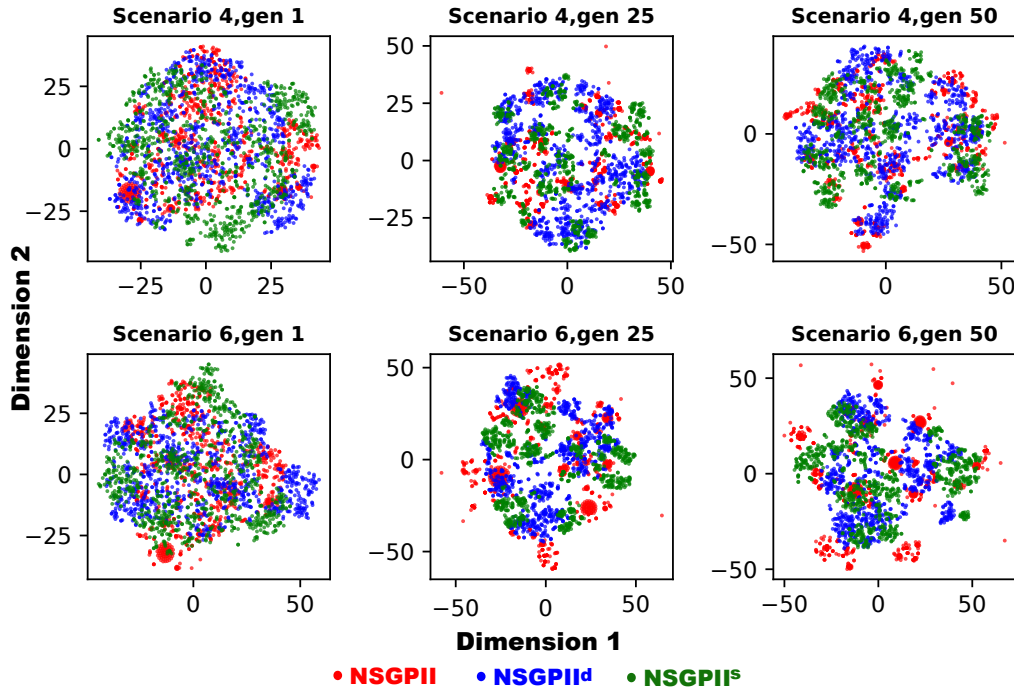


Figure 6.3: Visualisations of the semantics of individuals of one run of NSGP^{II}, NSGP^{II d}, and NSGP^{II s} on the scenario 4 and scenario 6 during the start (generation 1), middle (generation 25), and late (generation 50) stages of evolution.

The proposed semantic diversity and semantic similarity strategies aim

to limit the semantic distance between individuals in the population. It is interesting to study the semantic distribution of individuals in the population. The semantic represents the behaviour of the individual, which is a 40-dimensional vector. To visualise the semantics of individuals in the population, we employ t-SNE to reduce the dimensions to a 2-dimensional space.

Specifically, Figure 6.3 visualises the dimensionality reduced semantic of individuals in the population of NSGP_{II}, NSGP_{II}^d, and NSGP_{II}^s across different generations (1, 25, and 50) in scenarios 4 and 6. From Figure 6.3, we can clearly see that NSGP_{II} has several regions of more concentrated semantic distribution in each subfigure. This aligns with our expectations, as NSGP_{II} does not impose limitations on semantic distance between individuals. Compared to NSGP_{II}, the semantic distributions obtained by NSGP_{II}^s, on the other hand, are relatively widespread and do not have as clearly concentrated areas as NSGP_{II}. Compared to NSGP_{II} and NSGP_{II}^s, NSGP_{II}^d gives the most diverse semantic distributions. This finding highlights the significance of restricting the semantic distance between individuals, as it allows to achieve a population with a relatively more uniform semantic distribution, avoiding losing diversity, and potentially leading to better final scheduling heuristics. These insights emphasise the importance of controlling semantic distances between individuals during the evolutionary process of NSGP_{II}. Furthermore, it reveals that the improved final performance achieved by the inclusion of semantic information in NSGP_{II} is attributed to its ability to evolve a more semantically diverse population.

6.6 Chapter Summary

This chapter first proposes a MOGP/D method for the MO-DFJSS problem, aiming to evolve a Pareto front of scheduling heuristics capable of handling multiple objectives simultaneously. Through conducted exper-

iments, it is evident that, in comparison to the existing state-of-the-art multi-objective GP algorithm for DFJSS (NSGP-II), MOGP/D exhibits improved spreadability of trained Pareto front of scheduling heuristics on unseen test instances. However, MOGP/D cannot achieve better HV and IGD performance than NSGP-II. Consequently, this chapter further proposes a novel approach to improve the NSGP-II, namely the semantic NSGP-II method, designed to integrate semantic information into NSGP-II. This study successfully demonstrates the effectiveness of the proposed semantic NSGP-II in evolving a Pareto front of scheduling heuristics for effectively addressing the MO-DFJSS problem. To be specific, this study first contributes to giving the definitions of the semantic and semantic distance of scheduling heuristics for DFJSS. Then, by incorporating semantic diversity and semantic similarity within NSGP-II, this study contributes to evolving better scheduling heuristics than using the original NSGP-II.

The results highlight the benefits of considering semantically diverse individuals for achieving high-quality scheduling heuristics. Moreover, NSGP-II, considering semantic similarity, achieves the best overall performance, offering valuable insight into the importance of maintaining a reasonable semantic distance between offspring and their parents to further enhance the quality of scheduling heuristics. This emphasises the trade-off between semantic diversity and semantic similarity. Furthermore, the analysis of the population semantic distribution reveals that by controlling semantic distances between individuals, we are able to achieve a more semantically diverse population. This is the key factor contributing to the enhanced performance achieved by the proposed methods. Overall, this chapter demonstrates the potential of incorporating semantic information into the evolution process of NSGP-II for MO-DFJSS, providing valuable insights into the benefits and considerations of utilising semantic information in solving complex scheduling problems.

Chapter 7

Conclusions

This thesis focuses on addressing the challenges of DFJSS through the development of innovative GP methods. The overall goal is to improve the capability of GP to automatically learn effective scheduling heuristics for DFJSS. This goal has been successfully achieved through a comprehensive exploration of various aspects of GP, including the integration of diversity-based parent selection mechanisms, the establishment of stable and reliable joint decision-making via ensemble, the infusion of intelligent heuristic selection via RL, and the enhancement of multi-objective problem-solving capabilities through novel multi-objective GP algorithms. The effectiveness of each proposed algorithm is measured using the DFJSS simulation with a range of measurements and analyses.

The rest of this chapter highlights the achieved objectives in this thesis, followed by the main conclusions. This is followed by an in-depth discussion that illustrates the key insights from this research area. Finally, potential directions for future research are outlined, drawing from the findings and contributions of this thesis.

7.1 Achieved Objectives

This thesis has achieved the following research objectives.

1. **Firstly**, this thesis has proposed three novel diversity-based parent selection mechanisms for GP, enhancing its capability to explore solution spaces and evolving high-quality scheduling heuristics effectively (Chapter 3). These mechanisms, namely cluster selection, innovative diverse partner selection, and new lexicase selection, have demonstrated remarkable efficacy. The new cluster selection operator adeptly chooses two parents with distinct behaviours for crossover, showcasing a noteworthy phenomenon: the crossover between parents with dissimilar behaviours increases the number of unique behaviours within the population. The innovative diverse partner selection operator effectively selects a pair of parents with high quality and complements each other for crossover. Meanwhile, the new lexicase selection operator excels at choosing parents with expertise in different cases. To use the newly designed diverse partner selection and lexicase selection mechanisms efficiently, a new multi-case fitness evaluation strategy is developed. The multi-case fitness evaluation strategy divides a large DFJSS simulation into multiple cases to extract a list of case-fitnesses. This is more efficient than directly evaluating individuals on multiple instances. The effectiveness of the proposed GP methods with these parent selection operators is verified through comparisons with various existing GP methods across multiple DFJSS scenarios. Further analysis reveals that the increased population diversity by these diversity-based parent selection operators significantly contributes to the generation of high-quality scheduling heuristics for DFJSS.
2. **Secondly**, this thesis has developed a novel ensemble GP method to make good and trustworthy joint decisions for DFJSS (Chapter 4). The proposed ensemble GP introduces a specific population structure comprising both single individuals and ensembles. This method enables the evolution of either a single scheduling heuristic for individual decisions or a group of scheduling heuristics for

joint decisions within a single population in solving the DFJSS problem. In addition, the proposed ensemble construction and selection strategy, coupled with innovative crossover and mutation operators, contribute to an effective exploration of the search space through interbreeding. Experimental results showcase the superiority of this method over existing traditional and ensemble GP methods, attributing success to good and trustworthy joint decision-making, enhanced population diversity, and extensive search space exploration.

3. **Thirdly**, this thesis has explored an integration of GP and RL for making intelligent heuristics usage into the DFJSS domain (Chapter 5). Initially, a comparative analysis between a typical GP method and a typical RL method in DFJSS is conducted. After evaluating the strengths and limitations of GP and RL, a novel niching GP-assisted RL method is proposed. This method intelligently selects scheduling heuristics evolved by the niching GP for decision-making at various decision points. It replaces manual scheduling heuristics with those evolved by the niching GP, with RL optimising and adapting scheduling heuristics based on real-time feedback. Experimental results confirm the effectiveness of the proposed method, outperforming traditional methods and highlighting the behavioural distinctions among learned scheduling heuristics by the niching GP and their contributions throughout the scheduling process. Although this integration does not improve GP's performance on the DFJSS problem, it successfully extends GP's capability to play a significant role in improving RL's performance on DFJSS.
4. **Lastly**, this thesis has presented two multi-objective GP methods for evolving a Pareto front of high-quality scheduling heuristics to tackle the MO-DFJSS problem (Chapter 6). The first method combines a well-known multi-objective evolutionary algorithm based on decomposition with GP, enhancing the spreadability of the Pareto

front of scheduling heuristics. The second method enhances an existing state-of-the-art multi-objective GP method (NSGP-II) by integrating semantic information. For this method, the concepts of *semantic* and *semantic distance* are first defined within the context of the MO-DFJSS domain. Then, strategies are proposed to measure the semantic information obtained from MO-DFJSS. Subsequently, semantic NSGP-II methods considering semantic diversity and semantic similarity are developed. The experiment results demonstrate that NSGP-II methods considering semantic diversity and semantic similarity yield better performance compared with the original NSGP-II through effective control of semantic distances between individuals.

In summary, this thesis provides a comprehensive and diverse set of contributions, advancing the state-of-the-art in GP methods for DFJSS and paving the way for further research and exploration in this domain.

7.2 Main Conclusions

In this section, the main conclusions of this thesis drawn from each contribution chapter (i.e., Chapter 3 to 6) are described.

7.2.1 Diversity-based Parent Selection Mechanisms

To select suitable parents for generating high-quality offspring in GP, three main issues are required to be considered. Firstly, is relying solely on fitness as the selection criterion sufficient, or would incorporating additional criteria such as diversity enhance the effectiveness of parent selection? Secondly, when diversity is considered, how should the difference between individuals be measured? Lastly, after determining the diversity among individuals, how can suitable individuals be selected as parents?

Chapter 3 introduces two strategies for measuring the differences between individuals. The first strategy involves a phenotypic characteristic-

based method, wherein individuals are mapped to a list of values representing their behaviour. The distance between these lists serves as a straightforward measure of the difference between individuals. The second strategy is the proposed multi-case fitness-based method, which efficiently compiles a list of case-fitnesses from a single instance. Each case-fitness signifies an individual's performance on a subset of jobs within this instance. The differences between these case-fitnesses are utilised to measure the differences between individuals. Chapter 3 proposes three novel diversity-based parent selection mechanisms: cluster selection, diverse partner selection, and lexicase selection. These mechanisms facilitate the selection of individuals as parents based on not only fitness but also their distinctive behaviours, allowing each to contribute their own strength to the generation of high-quality offspring. The success of these parent selection mechanisms lies in their ability to choose high-quality individuals with diverse behaviours as parents, thereby generating offspring that exhibit both quality and diversity within the population.

The results show that the proposed GP algorithms, incorporating novel diversity-based parent selection mechanisms, effectively choose high-quality and diverse parents, leading to the generation of high-quality offspring. The algorithms also demonstrate an enhanced population diversity and a better balance between exploration and exploitation. Furthermore, the newly developed multi-case fitness definition is expected to be extended beyond DFJSS, providing general guidance for efficiently using these novel parent selection mechanisms to solve other intricate problems with time-consuming fitness evaluations, such as dynamic vehicle routing and cloud resource allocation.

7.2.2 Joint Decision Making with Ensemble

To make good and reliable decisions, using an ensemble of multiple scheduling heuristics to make joint decisions is an effective way, while two

issues need to be considered. Firstly, how to construct an ensemble using suitable individuals as its components? Secondly, whether the constructed ensembles be further enhanced through some evolutionary mechanisms?

Chapter 4 proposes an effective ensemble construction and selection strategy for GP. This strategy focuses on selecting individuals with high-quality and complementary characteristics to form an ensemble. The resulting ensemble is utilised to facilitate good and reliable joint decision-making at decision points. Furthermore, the chapter presents a novel population framework that allows both individuals and ensembles to evolve together within a single population. Additionally, new genetic operators (crossover and mutation) are developed to enable the breeding of individuals and ensembles, generating offspring that can be either individuals or ensembles. The success of the proposed algorithm can be attributed to the careful selection of high-quality and complementary individuals for ensemble formation. Furthermore, the effective evolutionary mechanisms, allowing individuals and ensembles to breed together, contribute to exploring the search space more comprehensively.

Comprehensive experiments and analyses demonstrate the effectiveness of the proposed method, particularly in terms of the quality of evolved scheduling heuristics compared to existing popular GP methods. Precisely, the proposed method demonstrates a noteworthy improvement in test performance, boasting a 0.73% enhancement, and achieves a commendable 8.72% reduction in training time compared to classic GP methods. The effectiveness of key strategies, such as ensemble construction and selection, along with genetic operators accommodating both single individuals and ensembles, has been verified, contributing to the overall improved performance of the proposed method. Further analyses reveal that these strategies foster the generation of high-quality single individuals and ensembles by preserving population diversity and encouraging high ensemble contributions from individual elements. Moreover, structural analyses of ensemble elements indicate that a good ensemble con-

tains elements with both shared subtrees and distinctive subtrees, allowing for effective complementarity and finally leading to improved overall decision-making capabilities. This combination of the ensemble enables the ensemble to make reliable decisions for a wide range of decision points while also excelling in specific decision points. Overall, these findings advance the understanding of GP and ensemble learning, and the proposed method holds promising applications in real-world scheduling scenarios.

7.2.3 Intelligent Heuristic Generation and Selection by GP and RL

To make an effective schedule that addresses numerous decision points, a possible way is to intelligently select a suitable scheduling heuristic at each decision point. This process involves addressing two primary challenges. First, there is a need to generate a diverse set of scheduling heuristics with distinct behaviours. Second, the task involves developing a mechanism for intelligent decision-making when faced with a decision point.

Chapter 5 presents a two-stage framework that integrates GP and RL for learning effective sequencing and routing agents to address the DFJSS problem. Specifically, the niching GP method is employed to learn a diverse set of high-quality scheduling heuristics. Subsequently, the sequencing rules derived from these learned scheduling heuristics serve as actions for the DRL method. This method tackles the challenge of adapting to altered operations at various decision points, enabling the learning of intelligent agents capable of making optimal selections among these actions to generate effective schedules.

Comparative results against baseline DRL and widely used manually designed scheduling heuristics validate the effectiveness of the proposed method. Additionally, comparisons against traditional GP-assisted DRL confirm the effectiveness of the proposed niching GP method. Furthermore, contrasting results against the proposed method without the DRL

training process, where the sequencing rule is fixed as one of the candidate actions, verify the efficacy of the DRL learning process. Further analysis of action contribution demonstrates that the scheduling heuristics learned by the niching GP method contribute similar percentages to the overall performance. The behavioural analysis reveals that the proposed niching GP method can learn diverse scheduling heuristics compared to traditional GP methods. Structural analysis indicates that the learned scheduling heuristics by the niching GP exhibit similarities but also show distinct performances at different decision points. While this integration of GP and RL does not enhance GP's performance on the DFJSS problem, it effectively extends GP's capability to significantly improve RL's performance. Future work could further investigate more effective combinations of GP and RL to enhance the capabilities of both GP and RL for DFJSS.

7.2.4 Multiple Objectives-solving Ability in GP

To improve the multiple objectives-solving ability of GP, two main issues need to be considered. Firstly, how to evolve a Pareto front of scheduling heuristics with good spreadability and high consistency? Secondly, how to consider semantic information in the evolution process to generate a Pareto front consisting of high-quality scheduling heuristics?

Chapter 6 proposes two novel multi-objective GP algorithms to evolve a high-quality Pareto front. The first algorithm, the multi-objective GP based on decomposition (MOGP/D), decomposes the MO-DFJSS problem into scalar sub-problems, optimising them simultaneously in a single run using different combinations of weights. The second algorithm, the semantic NSGP-II, introduces the concepts of *semantic* and *semantic distance* in the MO-DFJSS domain. Semantic diversity and semantic similarity strategies are then developed, with the former accepting individuals with unique semantics as offspring, and the latter further constraining the semantic distance between offspring and their parents. Finally, two en-

hanced NSGP-II algorithms are proposed by integrating NSGP-II with either the semantic diversity strategy or the semantic similarity strategy.

The results show that the proposed algorithms can evolve high-quality Pareto fronts for MO-DFJSS. Although MOGP/D exhibits inferior performance in terms of HV and IGD metrics, it demonstrates superior spreadability compared to all other methods. Furthermore, the results effectively show the benefits of integrating semantic information into NSGP-II for tackling the MO-DFJSS problem. Firstly, this study contributes to giving the definitions of the semantic and semantic distance of scheduling heuristics for DFJSS. Then, by incorporating semantic diversity and semantic similarity within NSGP-II, this study contributes to evolving better scheduling heuristics than using the original NSGP-II. The results highlight the benefits of considering semantically diverse individuals for achieving high-quality scheduling heuristics. Moreover, NSGP-II, considering semantic similarity, achieves the best overall performance, offering valuable insight into the importance of maintaining a reasonable semantic distance between offspring and their parents to further enhance the quality of scheduling heuristics. This emphasises the trade-off between semantic diversity and semantic similarity. Furthermore, the analysis of the population semantic distribution reveals that by controlling semantic distances between individuals, it is able to achieve a more semantically diverse population. This is the key factor contributing to the enhanced performance achieved by the proposed methods.

7.2.5 Overall Contributions to GP and AI

This thesis proposes advancements to GP on four key aspects. These advancements enhance its ability to learn high-quality scheduling heuristics specifically for solving the DFJSS problem. However, the contributions of this thesis extend beyond the DFJSS domain and can be applied to other artificial intelligence domains.

Firstly, in GP, maintaining a diverse population is crucial for effective exploration of the search space and preventing premature convergence. By proposing new diversity-based parent selection mechanisms, this thesis addresses a fundamental challenge in GP, encouraging the exploration of a wider range of possibilities, leading to potentially more creative solutions. Secondly, this thesis incorporates ensemble learning within GP. The joint decision making by ensemble GP can leverage the strengths of different individual models within the ensemble to create a more robust solution. Moreover, the proposed ensemble GP method opens doors for further research on allowing flexible breeding between single individuals and ensembles more deeply within the GP framework. Thirdly, by proposing the NichGPDL for intelligent heuristic generation and selection using a combination of GP and RL, this thesis leverages the strengths of both GP and RL, using GP's ability to generate a diverse set of heuristics with RL's focus on selecting the best heuristic for specific situations. The idea of hybridising GP and RL opens doors for further research on integrating GP and RL. Lastly, by proposing two new multi-objective methods, this thesis contributes to the field of multi-objective optimisation within GP. The proposed methods expand the applicability of GP to tackle complex problems with multiple, potentially conflicting objectives and encourage further exploration of how GP can be effectively used for multi-objective problem solving by using semantic information. In conclusion, all the proposed methods in this thesis can be adaptable to different problem types by incorporating domain-specific knowledge into the selection process.

7.3 Future Work

7.3.1 Training Time Reduction

Reducing the training time of GP is an important and growing research topic in the field of evolutionary algorithms. Since GP methods involve the

iterative evolution of programs, the computational demands of training, particularly in domains like DFJSS with a considerable number of jobs, can be very large. While this thesis does not explicitly focus on reducing training time, it touches upon this aspect in two studies. Firstly, to mitigate the risk of increasing training time, a multi-case fitness evaluation strategy (Chapter 3) is proposed in this thesis when employing the newly designed diverse partner selection and lexicase selection. This strategy allows for efficient evaluations by dividing one DFJSS instance into multiple cases instead of directly using multiple instances, minimising the impact on training time. Secondly, in the context of ensemble-based decision-making (Chapter 4), this thesis contributes to time reduction by reducing the warm-up process and maintaining a constant population size (number of scheduling heuristics), whether they come from individuals or ensembles. This method simplifies the evaluation and potentially saves training time. While these efforts address specific aspects of training time, a more focused exploration of strategies explicitly aimed at reducing GP training time remains an area for future research. Researchers can potentially combine various methods to achieve more substantial reductions in training time, considering the unique challenges posed by different problem domains.

The following are some aspects and potential future directions related to the topic of reducing training time in GP:

1. **Efficient fitness evaluation:** Optimising the fitness evaluation step is crucial. Researchers can investigate techniques to streamline the evaluation of candidate solutions, such as employing surrogate models, approximation methods, or pre-selection approaches to identify promising candidates for more thorough evaluation.
2. **Smart initialisation and warm starting:** Developing intelligent initialisation strategies to provide a good starting point for evolution can aid in quicker convergence. Warm-starting approaches, where

the evolution is initialised with information from previous runs, can be investigated to accelerate the convergence process.

3. **Algorithmic improvements:** Investigating algorithmic enhancements, such as more efficient crossover and mutation operators, can contribute to faster convergence. Designing operators that are tailored to the problem structure and encourage rapid exploration can be beneficial.

7.3.2 Interpretability of Scheduling Heuristics

Interpretability of scheduling heuristics evolved by GP is a critical research topic, gaining prominence as artificial intelligence systems become more prevalent in practical applications. Understanding the decision-making processes of evolved scheduling heuristics is crucial for their acceptance and practical implementation. GP, with its symbolic and tree-based representation, has an inherent interpretability that aligns with how humans understand the structure of computer programs [142]. Based on existing studies and a survey on the interpretability of GP, a GP model is generally more interpretable if it possesses one or more of the following properties [142]: 1) A GP model with fewer nodes in its tree-based representation requires fewer steps for a human user to simulate its behaviour, thus improving simulatability. 2) A GP model with simpler functions and a flatter, shallower tree structure also enhances simulatability. 3) A GP model that uses fewer distinct features contains fewer concepts to comprehend, making it easier to interpret. 4) A GP model with structures that follow certain grammars enhances decomposability and simulatability. This thesis focuses on improving the performance of GP in solving the DFJSS problems. It also touches on the topic of interpretability by analysing the tree size and structure of the evolved scheduling heuristics. While the present thesis may not delve deeply into interpretability, future research in this domain can explore several aspects:

1. **Explanation mechanisms for evolved heuristics:** Develop techniques or frameworks that generate human-readable explanations for the decisions made by evolved scheduling heuristics. This could involve identifying critical features or decision points within the evolved programs and translating them into understandable rules or patterns.
2. **Visualisations:** Create visualisation tools that can represent the decision-making process of evolved scheduling heuristics in an intuitive and accessible manner. This could involve graphical representations of the scheduling logic or interactive interfaces that allow users to explore the behaviour of evolved heuristics.
3. **Human-in-the-loop approaches:** Investigate approaches that involve human feedback in the interpretability process. This could include methods where users provide input on the interpretability of evolved scheduling heuristics, helping to refine the evolved scheduling heuristics iteratively.

7.3.3 Effective Utilisation of Multiple Scheduling Heuristics

Effectively utilising multiple scheduling heuristics evolved by GP, rather than relying solely on the best one, represents a crucial and challenging research topic. In practical scenarios, a single “best” heuristic may not always be optimal for diverse and dynamic situations. Leveraging the diversity of evolved heuristics has the potential to yield more robust and adaptable scheduling solutions. This thesis addresses this aspect through two key studies. Firstly, the ensemble work (Chapter 4) exploits the strengths of multiple scheduling heuristics to make more informed joint decisions, revealing the advantages of using diverse heuristics for improved schedules. Secondly, the integration of GP and RL (Chapter 5) contributes to the

generation and selection of multiple scheduling heuristics to make decisions when meeting different decision points. While these studies demonstrate the potential of leveraging multiple scheduling heuristics, a more in-depth exploration in this area is necessary. Future research should concentrate on developing advanced strategies and methodologies that fully exploit the diversity and adaptability offered by multiple scheduling heuristics evolved through GP.

The following are examples of future work related to the topic of effective utilisation of multiple scheduling heuristics in GP:

1. **Ensemble learning strategies and decision fusion techniques:** Develop advanced ensemble learning strategies that can intelligently combine the outputs of multiple evolved heuristics. This could involve dynamic weighting, where the influence of each heuristic is adjusted based on the characteristics of the current scheduling instance or its historical performance. This could also involve exploring decision fusion techniques to integrate the decisions made by different evolved heuristics. Methods like voting systems, stacking, or hierarchical decision-making can be investigated to exploit the strengths of individual heuristics in various decision scenarios.
2. **Context-aware and adaptive scheduling mechanisms:** Investigate context-aware scheduling approaches that consider the specific context of the scheduling problem. Evolved heuristics could be chosen based on the characteristics of the jobs, machines, or the historical performance of heuristics in similar contexts. Design mechanisms that dynamically select and adapt scheduling heuristics based on the characteristics of the jobs, machines, or the historical performance of heuristics in similar contexts.

7.3.4 Advanced Multi/many-objective Optimisation

Advanced multi/many-objective optimisation by GP represents a significant and forward-looking research topic. This thesis contributes to this field through two studies. Firstly, a novel multi-objective GP based on the decomposition method is introduced, showcasing its ability to successfully evolve a Pareto front with good spreadability and consistency. Secondly, an enhanced NSGPII incorporating semantic information is proposed, demonstrating its effectiveness in evolving high-quality Pareto fronts. While existing techniques have made progress, there is room for further advancements to address the complexities of real-world problems with multiple conflicting objectives. Future research in this area should focus on developing advanced methods within the context of GP. Here are some examples of potential future work:

1. **Hybrid approaches:** Investigate hybrid approaches that combine GP with other machine learning techniques, such as surrogate or transfer learning, to enhance its capability of solving multi/many-objective problems.
2. **Preference-based multi-objective optimisation:** Integrate preference-based methods into GP that can efficiently handle preference information for multi-objective optimisation. This involves incorporating human preferences into the optimisation process to guide the algorithm towards solutions that align with decision-makers' subjective preferences.

7.3.5 Integration of GP and RL

Considering the strengths and limitations of GP and RL, integrating them to enhance the capabilities of both for addressing JSS problems offers a promising approach to tackle complex optimisation challenges in manufacturing and production environments. This thesis touches upon this as-

pect in one study (Chapter 5). The integration of GP and RL contributes to the generation and selection of multiple scheduling heuristics for solving the DFJSS problem. While this study verifies the effectiveness of the integration of GP and RL over baseline RL and manually designed scheduling heuristics, the results cannot demonstrate its effectiveness over baseline GP. Future research could focus on developing more effective integration methods that enhance the capabilities of both GP and RL. Here are some potential avenues for future work:

1. **Considering both global and local feedback:** In GP, the optimisation objective (fitness function) typically aligns with the overarching optimisation goal and is determined after the entire scheduling process, relying on global information. Conversely, RL utilises an optimisation objective (reward function) that often differs from the overall goal, being provided based on local information. This distinction paves the way for a hybrid GP and RL approach that effectively integrates both global and local feedback.
2. **Experience information utilisation:** In RL, historical experience is sometimes stored in a replay memory to aid future decisions. GP can adopt this concept by storing experience information, which can then guide the population breeding process. For example, this information can inform decisions regarding crossover or mutation to improve scheduling heuristics.
3. **Attention mechanisms:** Certain RL methods utilise attention mechanisms to prioritise important information while disregarding noise. GP can integrate similar mechanisms to selectively focus on dominant information and eliminate the influence of redundant or noisy data from the training instances that could potentially degrade performance.
4. **Population mechanisms:** GP operates by maintaining and optimising a population of scheduling heuristics concurrently. These heuris-

tics explore the search space from various regions simultaneously, fostering diversity. RL can leverage this concept by optimising multiple scheduling heuristics simultaneously, allowing them to reinforce each other and collectively contribute to improved performance.

Bibliography

- [1] ADIBI, M., ZANDIEH, M., AND AMIRI, M. Multi-objective scheduling of dynamic job shop using variable neighborhood search. *Expert Systems with Applications* 37, 1 (2010), 282–287.
- [2] AENUGU, S., AND SPECTOR, L. Lexicase selection in learning classifier systems. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2019), pp. 356–364.
- [3] ARDEH, M. A., MEI, Y., AND ZHANG, M. Transfer learning in genetic programming hyper-heuristic for solving uncertain capacitated arc routing problem. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2019), pp. 49–56.
- [4] ARTIGUES, C., GENDREAU, M., ROUSSEAU, L. M., AND VERGNAUD, A. Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound. *Computers & Operations Research* 36, 8 (2009), 2330–2340.
- [5] ASHLOCK, D. *Evolutionary computation for modeling and optimization*, vol. 571. Springer, 2006.
- [6] ASHOUR, S., AND HIEMATH, S. A branch-and-bound approach to the job-shop scheduling problem. *International Journal of Production Research* 11, 1 (1973), 47–58.

- [7] ASLAM, M. W., ZHU, Z., AND NANDI, A. K. Diverse partner selection with brood recombination in genetic programming. *Applied Soft Computing* 67 (2018), 558–566.
- [8] AYDIN, M. E., AND FOGARTY, T. C. A simulated annealing algorithm for multi-agent systems: a job-shop scheduling application. *Journal of Intelligent Manufacturing* 15 (2004), 805–814.
- [9] AZEM, S., AGGOUNE, R., AND DAUZÈRE PÉRÈS, S. Disjunctive and time-indexed formulations for non-preemptive job shop scheduling with resource availability constraints. In *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management* (2007), pp. 787–791.
- [10] AZIZ, Z. A., ABDULQADER, D. N., SALLOW, A. B., AND OMER, H. K. Python parallel processing and multiprocessing: A review. *Academic Journal of Nawroz University* 10, 3 (2021), 345–354.
- [11] BÄCK, T., AND SCHWEFEL, H.-P. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation* 1, 1 (1993), 1–23.
- [12] BACK, T., AND SCHWEFEL, H. P. Evolutionary computation: An overview. In *Proceedings of IEEE International Conference on Evolutionary Computation* (1996), pp. 20–29.
- [13] BAI, H., CHENG, R., AND JIN, Y. Evolutionary reinforcement learning: A survey. *Intelligent Computing* 2 (2023). DOI:[10.34133/icomputing.0025](https://doi.org/10.34133/icomputing.0025).
- [14] BAI, R., BURKE, E. K., AND KENDALL, G. Heuristic, meta-heuristic and hyper-heuristic approaches for fresh produce inventory control and shelf space allocation. *Journal of the Operational Research Society* 59, 10 (2008), 1387–1397.

- [15] BAKUROV, I., CASTELLI, M., FONTANELLA, F., DI FRECA, A. S., AND VANNESCHI, L. A novel binary classification approach based on geometric semantic genetic programming. *Swarm and Evolutionary Computation* 69 (2022), 101028.
- [16] BANZHAF, W., NORDIN, P., KELLER, R. E., AND FRANCONI, F. D. *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. Morgan Kaufmann Publishers Inc., 1998.
- [17] BARLOW, H. B. Unsupervised learning. *Neural Computation* 1, 3 (1989), 295–311.
- [18] BATISTA, J. E., AND SILVA, S. Comparative study of classifier performance using automatic feature construction by m3gp. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2022), pp. 1–8. DOI:[10.1109/CEC55065.2022.9870343](https://doi.org/10.1109/CEC55065.2022.9870343).
- [19] BEADLE, L., AND JOHNSON, C. G. Semantically driven crossover in genetic programming. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2008), pp. 111–116.
- [20] BEADLE, L., AND JOHNSON, C. G. Semantic analysis of program initialisation in genetic programming. *Genetic Programming and Evolvable Machines* 10, 3 (2009), 307–337.
- [21] BEADLE, L., AND JOHNSON, C. G. Semantically driven mutation in genetic programming. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2009), pp. 1336–1342.
- [22] BERTSIMAS, D., AND TSITSIKLIS, J. Simulated annealing. *Statistical Science* 8, 1 (1993), 10–15.
- [23] BHOWAN, U., JOHNSTON, M., ZHANG, M., AND YAO, X. Evolving diverse ensembles using genetic programming for classification

- with unbalanced data. *IEEE Transactions on Evolutionary Computation* 17, 3 (2012), 368–386.
- [24] BI, Y., XUE, B., AND ZHANG, M. An effective feature learning approach using genetic programming with image descriptors for image classification. *IEEE Computational Intelligence Magazine* 15, 2 (2020), 65–77.
- [25] BIANCHI, R. A., RIBEIRO, C. H., AND COSTA, A. H. Accelerating autonomous learning by using heuristic selection of actions. *Journal of Heuristics* 14 (2008), 135–168.
- [26] BLICKLE, T. Tournament selection. *Evolutionary Computation* 1 (2000), 181–186.
- [27] BLICKLE, T., AND THIELE, L. A mathematical analysis of tournament selection. In *Proceedings of the International Conference on Genetic Algorithm* (1995), vol. 95, pp. 9–15.
- [28] BRAUNE, R., BENDA, F., DOERNER, K. F., AND HARTL, R. F. A genetic programming learning approach to generate dispatching rules for flexible shop scheduling problems. *International Journal of Production Economics* 243 (2022), 108342.
- [29] BRIZUELA, C. A., AND SANNOMIYA, N. A diversity study in genetic algorithms for job shop scheduling problems. In *Proceedings of the Genetic and Evolutionary Computation Conference* (1999), pp. 75–82.
- [30] BURKE, E. K., GENDREAU, M., HYDE, M., KENDALL, G., OCHOA, G., ÖZCAN, E., AND QU, R. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society* 64 (2013), 1695–1724.
- [31] BURKE, E. K., GUSTAFSON, S., AND KENDALL, G. Diversity in genetic programming: An analysis of measures and correlation with

- fitness. *IEEE Transactions on Evolutionary Computation* 8, 1 (2004), 47–62.
- [32] BURKE, E. K., HYDE, M., KENDALL, G., OCHOA, G., ÖZCAN, E., AND WOODWARD, J. R. A classification of hyper-heuristic approaches. *Handbook of Metaheuristics* (2010), 449–468.
- [33] BURKE, E. K., HYDE, M. R., KENDALL, G., AND WOODWARD, J. Automatic heuristic generation with genetic programming: evolving a jack-of-all-trades or a master of one. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2007), pp. 1559–1565.
- [34] ÇALIŞ, B., AND BULKAN, S. A research survey: review of ai solution strategies of job shop scheduling problem. *Journal of Intelligent Manufacturing* 26 (2015), 961–973.
- [35] CHAND, S., HUYNH, Q., SINGH, H., RAY, T., AND WAGNER, M. On the use of genetic programming to evolve priority rules for resource constrained project scheduling problems. *Information Sciences* 432 (2018), 146–163.
- [36] CHANG, J., YU, D., HU, Y., HE, W., AND YU, H. Deep reinforcement learning for dynamic flexible job shop scheduling with random job arrival. *Processes* 10, 4 (2022), 760.
- [37] CHANG, P. C., CHEN, S. H., ZHANG, Q., AND LIN, J. L. Moea/d for flowshop scheduling problems. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2008), pp. 1433–1438.
- [38] CHAUDHRY, I. A., AND KHAN, A. A. A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research* 23, 3 (2016), 551–591.
- [39] CHEN, C., JI, Z., AND WANG, Y. Nsga-ii applied to dynamic flexible job shop scheduling problems with machine breakdown. *Modern Physics Letters B* 32, 34n36 (2018), 1840111.

- [40] CHEN, H., CHU, C., AND PROTH, J. M. An improvement of the lagrangean relaxation approach for job shop scheduling: a dynamic programming method. *IEEE Transactions on Robotics and Automation* 14, 5 (1998), 786–795.
- [41] CHEN, H., DING, G., QIN, S., AND ZHANG, J. A hyper-heuristic based ensemble genetic programming approach for stochastic resource constrained project scheduling problem. *Expert Systems with Applications* 167 (2021), 114174.
- [42] CHEN, Q., XUE, B., AND ZHANG, M. Improving generalization of genetic programming for symbolic regression with angle-driven geometric semantic operators. *IEEE Transactions on Evolutionary Computation* 23, 3 (2018), 488–502.
- [43] CHEN, Q., XUE, B., AND ZHANG, M. Preserving population diversity based on transformed semantics in genetic programming for symbolic regression. *IEEE Transactions on Evolutionary Computation* 25, 3 (2020), 433–447.
- [44] CHEN, Q., ZHANG, M., AND XUE, B. Geometric semantic genetic programming with perpendicular crossover and random segment mutation for symbolic regression. In *Proceedings of the International Conference on Simulated Evolution and Learning* (2017), pp. 422–434.
- [45] CHEN, X., BAI, R., QU, R., DONG, J., AND JIN, Y. Deep reinforcement learning assisted genetic programming ensemble hyper-heuristics for dynamic scheduling of container port trucks. *IEEE Transactions on Evolutionary Computation* (2024). DOI:[10.1109/TEVC.2024.3381042](https://doi.org/10.1109/TEVC.2024.3381042).
- [46] CHONG, C. S., LOW, M. Y. H., SIVAKUMAR, A. I., AND GAY, K. L. A bee colony optimization algorithm to job shop scheduling. In *Proceedings of the Winter Simulation Conference* (2006), pp. 1954–1961.

- [47] CHU, T. H., NGUYEN, Q. U., AND O'NEILL, M. Semantic tournament selection for genetic programming based on statistical analysis of error vectors. *Information Sciences* 436 (2018), 352–366.
- [48] CUNNINGHAM, P., CORD, M., AND DELANY, S. J. Supervised learning. In *Machine learning techniques for multimedia: case studies on organization and retrieval*. Springer, 2008, pp. 21–49.
- [49] DABHI, V. K., AND CHAUDHARY, S. Empirical modeling using genetic programming: A survey of issues and approaches. *Natural Computing* 14 (2015), 303–330.
- [50] DAY, P., AND NANDI, A. K. Binary string fitness characterization and comparative partner selection in genetic programming. *IEEE Transactions on Evolutionary Computation* 12, 6 (2008), 724–735.
- [51] DE RAINVILLE, F. M., FORTIN, F. A., GARDNER, M. A., PARIZEAU, M., AND GAGNÉ, C. Deap: A python framework for evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2012), pp. 85–92.
- [52] DEB, K., PRATAP, A., AGARWAL, S., AND MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.
- [53] DIETTERICH, T. G. Ensemble methods in machine learning. In *Proceedings of the International Workshop on Multiple Classifier Systems* (2000), pp. 1–15.
- [54] DIVINE, G., NORTON, H. J., HUNT, R., AND DIENEMANN, J. A review of analysis and sample size calculation considerations for wilcoxon tests. *Anesthesia & Analgesia* 117, 3 (2013), 699–710.
- [55] DRUGAN, M. M. Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms. *Swarm and Evolutionary Computation* 44 (2019), 228–246.

- [56] DU, Y., LI, J., LI, C., AND DUAN, P. A reinforcement learning approach for flexible job shop scheduling problem with crane transportation and setup times. *IEEE Transactions on Neural Networks and Learning Systems* (2022), 1–15. DOI:[10.1109/TNNLS.2022.3208942](https://doi.org/10.1109/TNNLS.2022.3208942).
- [57] DUMITRESCU, D., LAZZERINI, B., JAIN, L. C., AND DUMITRESCU, A. *Evolutionary computation*. CRC press, 2000.
- [58] EL NAQA, I., AND MURPHY, M. J. *What is machine learning?* Springer, 2015.
- [59] ENGIN, O., AND GÜÇLÜ, A. A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems. *Applied Soft Computing* 72 (2018), 166–176.
- [60] FAN, H., XIONG, H., AND GOH, M. Genetic programming-based hyper-heuristic approach for solving dynamic job shop scheduling problem with extended technical precedence constraints. *Computers & Operations Research* 134 (2021), 105401.
- [61] FANG, H. L., ROSS, P., AND CORNE, D. *A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems*. Berlin, Heidelberg: University of Edinburgh, Department of Artificial Intelligence, 1993.
- [62] FANG, Y., AND LI, J. A review of tournament selection in genetic programming. In *International Symposium on Intelligence Computation and Applications* (2010), pp. 181–192.
- [63] FATTAHI, P., MESSI BIDGOLI, M., AND SAMOUEI, P. An improved tabu search algorithm for job shop scheduling problem through hybrid solution representations. *Journal of Quality Engineering and Production Optimization* 3, 1 (2018), 13–26.
- [64] FERREIRA, C. *Gene expression programming: mathematical modeling by an artificial intelligence*, vol. 21. Springer, 2006.

- [65] FILAR, J., AND VRIEZE, K. *Competitive Markov decision processes*. Springer Science & Business Media, 2012.
- [66] FOGEL, D. B. An overview of evolutionary programming. In *Evolutionary Algorithms* (1999), Springer, pp. 89–109.
- [67] FONTES, D. B., HOMAYOUNI, S. M., AND GONÇALVES, J. F. A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources. *European Journal of Operational Research* 306, 3 (2023), 1140–1157.
- [68] FREITAS, D., LOPES, L. G., AND MORGADO DIAS, F. Particle swarm optimisation: a historical review up to the current developments. *Entropy* 22, 3 (2020), 362.
- [69] GALVÁN, E., AND SCHOENAUER, M. Promoting semantic diversity in multi-objective genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2019), pp. 1021–1029.
- [70] GALVAN LOPEZ, E., CODY KENNY, B., TRUJILLO, L., AND KATTAN, A. Using semantics in the selection mechanism in genetic programming: a simple method for promoting semantic diversity. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2013), pp. 2972–2979.
- [71] GAO, K., CAO, Z., ZHANG, L., CHEN, Z., HAN, Y., AND PAN, Q. A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. *Journal of Automatica Sinica* 6, 4 (2019), 904–916.
- [72] GAO, K. Z., SUGANTHAN, P. N., TASGETIREN, M. F., PAN, Q. K., AND SUN, Q. Q. Effective ensembles of heuristics for scheduling flexible job shop problem with new job insertion. *Computers & Industrial Engineering* 90 (2015), 107–117.

- [73] GIL GALA, F. J., MENCÍA, C., SIERRA, M. R., AND VARELA, R. Evolving priority rules for on-line scheduling of jobs on a single machine with variable capacity over time. *Applied Soft Computing* 85 (2019), 105782.
- [74] GIL GALA, F. J., MENCÍA, C., SIERRA, M. R., AND VARELA, R. Learning ensembles of priority rules for online scheduling by hybrid evolutionary algorithms. *Integrated Computer-Aided Engineering* 28, 1 (2021), 65–80.
- [75] GIL GALA, F. J., SIERRA, M. R., MENCÍA, C., AND VARELA, R. Combining hyper-heuristics to evolve ensembles of priority rules for on-line scheduling. *Natural Computing* 21, 4 (2022), 553–563.
- [76] GIL GALA, F. J., SIERRA, M. R., MENCÍA, C., AND VARELA, R. Surrogate model for memetic genetic programming with application to the one machine scheduling problem with time-varying capacity. *Expert Systems with Applications* 233 (2023), 120916.
- [77] GIL GALA, F. J., URASEVIĆ, M., VARELA, R., AND JAKOBOVIĆ, D. Ensembles of priority rules to solve one machine scheduling problem in real-time. *Information Sciences* 634 (2023), 340–358.
- [78] GIL GALA, F. J., AND VARELA, R. Genetic algorithm to evolve ensembles of rules for on-line scheduling on single machine with variable capacity. In *Proceedings of the International Work-Conference on the Interplay Between Natural and Artificial Computation* (2019), pp. 223–233.
- [79] GLOVER, F., AND LAGUNA, M. *Tabu search*. Springer, 1998.
- [80] GOLDBERG, D. E., AND DEB, K. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, vol. 1. 1991, pp. 69–93.

- [81] GOMES, M. C., BARBOSA PÓVOA, A. P., AND NOVAIS, A. Q. Reactive scheduling in a make-to-order flexible job shop with re-entrant process and assembly: a mathematical programming approach. *International Journal of Production Research* 51, 17 (2013), 5120–5141.
- [82] GONÇALVES, J. F., DE MAGALHÃES MENDES, J. J., AND RESENDE, M. G. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research* 167, 1 (2005), 77–95.
- [83] GROMICHO, J. A., VAN HOORN, J. J., SALDANHA DA GAMA, F., AND TIMMER, G. T. Solving the job-shop scheduling problem optimally by dynamic programming. *Computers & Operations Research* 39, 12 (2012), 2968–2977.
- [84] GUI, Y., TANG, D., ZHU, H., ZHANG, Y., AND ZHANG, Z. Dynamic scheduling for flexible job shop using a deep reinforcement learning approach. *Computers & Industrial Engineering* 180 (2023), 109255.
- [85] GUO, X., WANG, X., AND WEI, Z. Moea/d with adaptive weight vector design. In *Proceedings of the International Conference on Computational Intelligence and Security* (2015), pp. 291–294.
- [86] HANSEN, N., ARNOLD, D. V., AND AUGER, A. Evolution strategies. *Springer Handbook of Computational Intelligence* (2015), 871–898.
- [87] HART, E., AND ROSS, P. A heuristic combination method for solving job-shop scheduling problems. In *Proceedings of the International Conference on Parallel Problem Solving from Nature* (1998), pp. 845–854.
- [88] HART, E., AND SIM, K. A hyper-heuristic ensemble method for static job-shop scheduling. *Evolutionary Computation* 24, 4 (2016), 609–635.
- [89] HEINONEN, J., AND PETTERSSON, F. Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem. *Applied Mathematics and Computation* 187, 2 (2007), 989–998.

- [90] HELMUTH, T., AND ABDELHADY, A. Benchmarking parent selection for program synthesis by genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2020), pp. 237–238.
- [91] HELMUTH, T., MCPHEE, N. F., AND SPECTOR, L. The impact of hyperselection on lexicase selection. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2016), pp. 717–724.
- [92] HELMUTH, T., SPECTOR, L., AND MATHESON, J. Solving uncompromising problems with lexicase selection. *IEEE Transactions on Evolutionary Computation* 19, 5 (2014), 630–643.
- [93] HILDEBRANDT, T., AND BRANKE, J. On using surrogates with genetic programming. *Evolutionary Computation* 23, 3 (2015), 343–367.
- [94] HILDEBRANDT, T., HEGER, J., AND SCHOLZ REITER, B. Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2010), pp. 257–264.
- [95] HO, N. B., AND TAY, J. C. Evolving dispatching rules for solving the flexible job-shop problem. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2005), vol. 3, pp. 2848–2855.
- [96] HOLLAND, J. H. Genetic algorithms. *Scientific American* 267, 1 (1992), 66–73.
- [97] HOLTHAUS, O., AND RAJENDRAN, C. Efficient dispatching rules for scheduling in a job shop. *International Journal of Production Economics* 48, 1 (1997), 87–105.
- [98] HUANG, J., GAO, L., AND LI, X. An end-to-end deep reinforcement learning method based on graph neural network for distributed

- job-shop scheduling problem. *Expert Systems with Applications* 238 (2024), 121756.
- [99] HUANG, K. L., AND LIAO, C. J. Ant colony optimization combined with taboo search for the job shop scheduling problem. *Computers & Operations Research* 35, 4 (2008), 1030–1046.
- [100] HUANG, R. H., AND YU, T. H. An effective ant colony optimization algorithm for multi-objective job-shop scheduling with equal-size lot-splitting. *Applied Soft Computing* 57 (2017), 642–656.
- [101] HUANG, V., WANG, C., MA, H., CHEN, G., AND CHRISTOPHER, K. Cost-aware dynamic multi-workflow scheduling in cloud data center using evolutionary reinforcement learning. In *Proceedings of the International Conference on Service Oriented Computing* (2022), Springer, pp. 449–464.
- [102] HUANG, Z., MEI, Y., ZHANG, F., AND ZHANG, M. Multitask linear genetic programming with shared individuals and its application to dynamic job shop scheduling. *IEEE Transactions on Evolutionary Computation* (2023). DOI:[10.1109/TEVC.2023.3263871](https://doi.org/10.1109/TEVC.2023.3263871).
- [103] HUNT, R. J., JOHNSTON, M. R., AND ZHANG, M. *Evolving dispatching rules with greater understandability for dynamic job shop scheduling*. Technical Report ECSTR15-06, School of Engineering and Computer Science, Victoria University of Wellington, 2015.
- [104] ISHIBUCHI, H., AKEDO, N., AND NOJIMA, Y. Relation between neighborhood size and moea/d performance on many-objective problems. In *Proceedings of the International Conference on Evolutionary Multi-criterion Optimization* (2013), Springer, pp. 459–474.
- [105] ISHIBUCHI, H., HITOTSUYANAGI, Y., OHYANAGI, H., AND NOJIMA, Y. Effects of the existence of highly correlated objectives on the behavior of moea/d. In *Proceedings of the International Conference on*

- Evolutionary Multi-Criterion Optimization* (2011), Springer, pp. 166–181.
- [106] ISHIBUCHI, H., AND SHIBATA, Y. Mating scheme for controlling the diversity-convergence balance for multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2004), pp. 1259–1271.
- [107] ISHIBUCHI, H., TANIGAKI, Y., MASUDA, H., AND NOJIMA, Y. Distance-based analysis of crossover operators for many-objective knapsack problems. In *Proceedings of the Parallel Problem Solving from Nature* (2014), pp. 600–610.
- [108] JIANG, E., WANG, L., AND WANG, J. Decomposition-based multi-objective optimization for energy-aware distributed hybrid flow shop scheduling with multiprocessor tasks. *Tsinghua Science and Technology* 26, 5 (2021), 646–663.
- [109] JIMÉNEZ LAREDO, J. L., NIELSEN, S. S., DANOY, G., BOUVRY, P., AND FERNANDES, C. M. Cooperative selection: improving tournament selection via altruism. In *Proceedings of the European Conference on Evolutionary Computation in Combinatorial Optimization* (2014), pp. 85–96.
- [110] JORDAN, M. I., AND MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. *Science* 349, 6245 (2015), 255–260.
- [111] KAEHLING, L. P., LITTMAN, M. L., AND MOORE, A. W. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4 (1996), 237–285.
- [112] KARUNAKARAN, D. *Active Learning Methods for Dynamic Job Shop Scheduling using Genetic Programming under Uncertain Environment*. PhD thesis, Open Access Te Herenga Waka-Victoria University of Wellington, 2019.

- [113] KARUNAKARAN, D., CHEN, G., AND ZHANG, M. Parallel multi-objective job shop scheduling using genetic programming. In *Proceedings of the Australasian Conference on Artificial Life and Computational Intelligence* (2016), pp. 234–245.
- [114] KARUNAKARAN, D., MEI, Y., CHEN, G., AND ZHANG, M. Evolving dispatching rules for dynamic job shop scheduling with uncertain processing times. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2017), pp. 364–371.
- [115] KARUNAKARAN, D., MEI, Y., CHEN, G., AND ZHANG, M. Toward evolving dispatching rules for dynamic job shop scheduling under uncertainty. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2017), pp. 282–289.
- [116] KARUNAKARAN, D., MEI, Y., CHEN, G., AND ZHANG, M. Sampling heuristics for multi-objective dynamic job shop scheduling using island based parallel genetic programming. In *Proceedings of International Conference on Parallel Problem Solving from Nature* (2018), pp. 347–359.
- [117] KENNEDY, J. Swarm intelligence. In *Handbook of nature-inspired and innovative computing: integrating classical models with emerging technologies*. Springer, 2006, pp. 187–219.
- [118] KHOZANI, Z. S., SAFARI, M. J. S., MEHR, A. D., AND MOHTAR, W. H. M. W. An ensemble genetic programming approach to develop incipient sediment motion models in rectangular channels. *Journal of Hydrology* 584 (2020), 124753.
- [119] KOZA, J. R. *Genetic programming II*, vol. 17. 1994.
- [120] KOZA, J. R., AND KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.

- [121] LA CAVA, W., HELMUTH, T., SPECTOR, L., AND MOORE, J. H. A probabilistic and multi-objective analysis of lexicase selection and ε -lexicase selection. *Evolutionary Computation* 27, 3 (2019), 377–402.
- [122] LA CAVA, W., AND MOORE, J. H. An analysis of ε -lexicase selection for large-scale many-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2018), pp. 185–186.
- [123] LA CAVA, W., SPECTOR, L., AND DANAI, K. Epsilon-lexicase selection for regression. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2016), pp. 741–748.
- [124] LEI, K., GUO, P., WANG, Y., ZHANG, J., MENG, X., AND QIAN, L. Large-scale dynamic scheduling for flexible job-shop with random arrivals of new jobs by hierarchical reinforcement learning. *IEEE Transactions on Industrial Informatics* 20, 1 (2024), 1007–1018.
- [125] LEI, K., GUO, P., ZHAO, W., WANG, Y., QIAN, L., MENG, X., AND TANG, L. A multi-action deep reinforcement learning framework for flexible job-shop scheduling problem. *Expert Systems with Applications* 205 (2022), 117796.
- [126] LI, H., AND ZHANG, Q. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE Transactions on Evolutionary Computation* 13, 2 (2008), 284–302.
- [127] LI, M., AND YAO, X. Quality evaluation of solution sets in multiobjective optimisation: A survey. *ACM Computing Surveys* 52, 2 (2019), 1–38.
- [128] LI, Y. Deep reinforcement learning: An overview. *ArXiv Preprint ArXiv:1701.07274* (2017).

- [129] LI, Y., GU, W., YUAN, M., AND TANG, Y. Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep q network. *Robotics and Computer-Integrated Manufacturing* 74 (2022), 102283.
- [130] LIAN, Z., JIAO, B., AND GU, X. A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan. *Applied Mathematics and Computation* 183, 2 (2006), 1008–1017.
- [131] LIN, C. C., DENG, D. J., CHIH, Y. L., AND CHIU, H. T. Smart manufacturing scheduling with edge computing using multiclass deep q network. *IEEE Transactions on Industrial Informatics* 15, 7 (2019), 4276–4284.
- [132] LIN, T. L., HORNG, S. J., KAO, T. W., CHEN, Y. H., RUN, R. S., CHEN, R. J., LAI, J. L., AND KUO, I. H. An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications* 37, 3 (2010), 2629–2636.
- [133] LIU, A., LUH, P. B., YAN, B., AND BRAGIN, M. A. A novel integer linear programming formulation for job-shop scheduling problems. *IEEE Robotics and Automation Letters* 6, 3 (2021), 5937–5944.
- [134] LIU, H., GU, F., AND ZHANG, Q. Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. *IEEE Transactions on Evolutionary Computation* 18, 3 (2013), 450–455.
- [135] LIU, K. C. Dispatching rules for stochastic finite capacity scheduling. *Computers & industrial engineering* 35, 1–2 (1998), 113–116.
- [136] LIU, R., PIPLANI, R., AND TORO, C. Deep reinforcement learning for dynamic scheduling of a flexible job shop. *International Journal of Production Research* 60, 13 (2022), 4049–4069.

- [137] LIU, Z., WANG, Y., LIANG, X., MA, Y., FENG, Y., CHENG, G., AND LIU, Z. A graph neural networks-based deep q-learning approach for job shop scheduling problems in traffic management. *Information Sciences* 607 (2022), 1211–1223.
- [138] LUKE, S. Ecj then and now. In *Proceedings of the Genetic and Evolutionary Computation Conference companion* (2017), pp. 1223–1230.
- [139] LUO, S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Applied Soft Computing* 91 (2020), 106208.
- [140] MASOOD, A., MEI, Y., CHEN, G., AND ZHANG, M. Many-objective genetic programming for job-shop scheduling. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2016), pp. 209–216.
- [141] MATTFELD, D. C., AND BIERWIRTH, C. An efficient genetic algorithm for job shop scheduling with tardiness objectives. *European Journal of Operational Research* 155, 3 (2004), 616–630.
- [142] MEI, Y., CHEN, Q., LENSEN, A., XUE, B., AND ZHANG, M. Explainable artificial intelligence by genetic programming: A survey. *IEEE Transactions on Evolutionary Computation* 27, 3 (2023), 621–641.
- [143] MEI, Y., NGUYEN, S., XUE, B., AND ZHANG, M. An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming. *IEEE Transactions on Emerging Topics in Computational Intelligence* 1, 5 (2017), 339–353.
- [144] MEI, Y., NGUYEN, S., AND ZHANG, M. Constrained dimensionally aware genetic programming for evolving interpretable dispatching rules in dynamic job shop scheduling. In *Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning* (2017), pp. 435–447.

- [145] MEI, Y., AND ZHANG, M. A comprehensive analysis on reusability of gp-evolved job shop dispatching rules. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2016), pp. 3590–3597.
- [146] MEI, Y., ZHANG, M., AND NGUYEN, S. Feature selection in evolving job shop dispatching rules with genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2016), pp. 365–372.
- [147] METEVIER, B., SAINI, A. K., AND SPECTOR, L. Lexicase selection beyond genetic programming. In *Genetic Programming Theory and Practice XVI*. 2019, pp. 123–136.
- [148] MILLER, B. L., AND GOLDBERG, D. E. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems* 9, 3 (1995), 193–212.
- [149] MIRJALILI, S., AND MIRJALILI, S. Ant colony optimisation. *Evolutionary Algorithms and Neural Networks: Theory and Applications* (2019), 33–42.
- [150] MIYASHITA, K. Job-shop scheduling with genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2000), pp. 505–512.
- [151] MONACI, M., AGASUCCI, V., AND GRANI, G. An actor-critic algorithm with deep double recurrent agents to solve the job shop scheduling problem. *arXiv preprint arXiv:2110.09076* (2021).
- [152] MOORE, J. M., AND STANTON, A. Tiebreaks and diversity: Isolating effects in lexicase selection. In *Proceedings of the Artificial Life Conference* (2018), pp. 590–597.
- [153] MORALES, E. F., AND ZARAGOZA, J. H. An introduction to reinforcement learning. In *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*. IGI Global, 2012, pp. 63–80.

- [154] NAG, K., AND PAL, N. R. A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification. *IEEE Transactions on Cybernetics* 46, 2 (2015), 499–510.
- [155] NERI, F., AND COTTA, C. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation* 2 (2012), 1–14.
- [156] NGUYEN, S., MEI, Y., XUE, B., AND ZHANG, M. A hybrid genetic programming algorithm for automated design of dispatching rules. *Evolutionary Computation* 27, 3 (2019), 467–496.
- [157] NGUYEN, S., MEI, Y., AND ZHANG, M. Genetic programming for production scheduling: a survey with a unified framework. *Complex & Intelligent Systems* 3, 1 (2017), 41–66.
- [158] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. A co-evolution genetic programming method to evolve scheduling policies for dynamic multi-objective job shop scheduling problems. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2012), pp. 1–8.
- [159] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. *IEEE Transactions on Evolutionary Computation* 17, 5 (2012), 621–639.
- [160] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. *IEEE Transactions on Evolutionary Computation* 18, 2 (2013), 193–208.
- [161] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. Dynamic multi-objective job shop scheduling: A genetic programming

- approach. *Automated scheduling and planning: from theory to practice* (2013), 251–282.
- [162] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. Selection schemes in surrogate-assisted genetic programming for job shop scheduling. In *Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning* (2014), pp. 656–667.
- [163] NGUYEN, S., ZHANG, M., JOHNSTON, M., AND TAN, K. C. Genetic programming for job shop scheduling. *Evolutionary and Swarm Intelligence Algorithms* (2019), 143–167.
- [164] NGUYEN, S., ZHANG, M., AND TAN, K. C. Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2015), pp. 2781–2788.
- [165] OZOLINS, A. Bounded dynamic programming algorithm for the job shop problem with sequence dependent setup times. *Operational Research* 20, 3 (2020), 1701–1728.
- [166] O'DONOGHUE, B., OSBAND, I., MUNOS, R., AND MNIH, V. The uncertainty bellman equation and exploration. In *Proceedings of the International Conference on Machine Learning* (2018), pp. 3836–3845.
- [167] PALACIOS ALONSO, J. J., AND DERBEL, B. On maintaining diversity in moea/d: Application to a biobjective combinatorial fjsp. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2015), pp. 719–726.
- [168] PANTRIDGE, E., HELMUTH, T., MCPHEE, N. F., AND SPECTOR, L. Specialization and elitism in lexicase and tournament selection. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2018), pp. 1914–1917.

- [169] PAPA, J. P., ROSA, G. H., AND PAPA, L. P. A binary-constrained geometric semantic genetic programming for feature selection purposes. *Pattern Recognition Letters* 100 (2017), 59–66.
- [170] PARK, J., CHUN, J., KIM, S. H., KIM, Y., AND PARK, J. Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning. *International Journal of Production Research* 59, 11 (2021), 3360–3377.
- [171] PARK, J., MEI, Y., CHEN, G., AND ZHANG, M. Niching genetic programming based hyper-heuristic approach to dynamic job shop scheduling: an investigation into distance metrics. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2016), pp. 109–110.
- [172] PARK, J., MEI, Y., NGUYEN, S., CHEN, G., JOHNSTON, M., AND ZHANG, M. Genetic programming based hyper-heuristics for dynamic job shop scheduling: Cooperative coevolutionary approaches. In *Proceedings of the European Conference on Genetic Programming* (2016), pp. 115–132.
- [173] PARK, J., MEI, Y., NGUYEN, S., CHEN, G., AND ZHANG, M. Evolutionary multitask optimisation for dynamic job shop scheduling using niched genetic programming. In *Proceedings of the Australasian Joint Conference on Artificial Intelligence* (2018), pp. 739–751.
- [174] PARK, J., MEI, Y., NGUYEN, S., CHEN, G., AND ZHANG, M. Investigating a machine breakdown genetic programming approach for dynamic job shop scheduling. In *Proceedings of the European Conference on Genetic Programming* (2018), pp. 253–270.
- [175] PARK, J., MEI, Y., NGUYEN, S., CHEN, G., AND ZHANG, M. An investigation of ensemble combination schemes for genetic pro-

- gramming based hyper-heuristic approaches to dynamic job shop scheduling. *Applied Soft Computing* 63 (2018), 72–86.
- [176] PARK, J., NGUYEN, S., ZHANG, M., AND JOHNSTON, M. Evolving ensembles of dispatching rules using genetic programming for job shop scheduling. In *Proceedings of the European Conference on Genetic Programming* (2015), pp. 92–104.
- [177] PARK, J., NGUYEN, S., ZHANG, M., AND JOHNSTON, M. A single population genetic programming based ensemble learning approach to job shop scheduling. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2015), pp. 1451–1452.
- [178] PASZKE, A. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems* 32 (2019).
- [179] PEZZELLA, F., MORGANTI, G., AND CIASCHETTI, G. A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research* 35, 10 (2008), 3202–3212.
- [180] PLANINIĆ, L., DURASEVIĆ, M., AND JAKOBOVIĆ, D. On the application of ϵ -lexicase selection in the generation of dispatching rules. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2021), pp. 2125–2132.
- [181] PONGCHAIRERKS, P., AND KACHITVICHYANUKUL, V. A two-level particle swarm optimisation algorithm on job-shop scheduling problems. *International Journal of Operational Research* 4, 4 (2009), 390–411.
- [182] POTTS, C. N., AND VAN WASSENHOVE, L. N. A branch and bound algorithm for the total weighted tardiness problem. *Operations Research* 33, 2 (1985), 363–377.

- [183] PUTERMAN, M. L. Markov decision processes. *Handbooks in Operations Research and Management Science* 2 (1990), 331–434.
- [184] QIAN, W., CHAI, J., XU, Z., AND ZHANG, Z. Differential evolution algorithm with multiple mutation strategies based on roulette wheel selection. *Applied Intelligence* 48 (2018), 3612–3629.
- [185] REN, W., YAN, Y., HU, Y., AND GUAN, Y. Joint optimisation for dynamic flexible job-shop scheduling problem with transportation time and resource constraints. *International Journal of Production Research* 60, 18 (2022), 5675–5696.
- [186] REN, Z., ZHU, G., HU, H., HAN, B., CHEN, J., AND ZHANG, C. On the estimation bias in double q-learning. *Advances in Neural Information Processing Systems* 34 (2021), 10246–10259.
- [187] RODRIGUES, N. M., BATISTA, J. E., AND SILVA, S. Ensemble genetic programming. In *Proceedings of the European Conference on Genetic Programming* (2020), pp. 151–166.
- [188] ROSS, H. L. F. P., AND CORNE, D. A promising hybrid ga/heuristic approach for open-shop scheduling problems. In *Proceedings of the European Conference on Artificial Intelligence* (1994), pp. 590–594.
- [189] SAGI, O., AND ROKACH, L. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8, 4 (2018), e1249.
- [190] SAIDI MEHRABAD, M., AND FATTAHI, P. Flexible job shop scheduling with tabu search algorithms. *International Journal of Advanced Manufacturing Technology* 32 (2007), 563–570.
- [191] SAINI, A. K., AND SPECTOR, L. Relationships between parent selection methods, looping constructs, and success rate in genetic programming. *Genetic Programming and Evolvable Machines* 22 (2021), 495–509.

- [192] SÁNCHEZ, C. N., AND GRAFF, M. Selection heuristics on semantic genetic programming for classification problems. *Evolutionary Computation* 30, 2 (2022), 253–289.
- [193] SARIN, S. C., AHN, S., AND BISHOP, A. B. An improved branching scheme for the branch and bound procedure of scheduling n jobs on m parallel machines to minimize total weighted flowtime. *International Journal of Production Research* 26, 7 (1988), 1183–1191.
- [194] SHADY, S., KAIHARA, T., FUJII, N., AND KOKURYO, D. A novel feature selection for evolving compact dispatching rules using genetic programming for dynamic job shop scheduling. *International Journal of Production Research* 60, 13 (2022), 4025–4048.
- [195] SHADY, S., KAIHARA, T., FUJII, N., AND KOKURYO, D. Feature selection approach for evolving reactive scheduling policies for dynamic job shop scheduling problem using gene expression programming. *International Journal of Production Research* 61, 15 (2023), 5029–5052.
- [196] SHANKER, K., AND TZEN, Y. J. J. A loading and dispatching problem in a random flexible manufacturing system. *International Journal of Production Research* 23, 3 (1985), 579–595.
- [197] SHARMA, P., AND JAIN, A. Stochastic dynamic job shop scheduling with sequence-dependent setup times: simulation experimentation. *Journal of Engineering and Technology* 5, 1 (2015), 19.
- [198] SINDHYA, K., MIETTINEN, K., AND DEB, K. A hybrid framework for evolutionary multi-objective optimization. *IEEE Transactions on Evolutionary Computation* 17, 4 (2012), 495–511.
- [199] SMART, W., AND ZHANG, M. Using genetic programming for multiclass classification by simultaneously solving component binary

- classification problems. In *Proceedings of the European Conference on Genetic Programming* (2005), pp. 227–239.
- [200] SOKOLOV, A., AND WHITLEY, D. Unbiased tournament selection. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2005), pp. 1131–1138.
- [201] SOKOLOV, A., WHITLEY, D., AND DA MOTTA SALLES BARRETO, A. A note on the variance of rank-based selection strategies for genetic algorithms and genetic programming. *Genetic Programming and Evolvable Machines* 8, 3 (2007), 221–237.
- [202] SPECTOR, L. Assessment of problem modality by differential performance of lexicase selection in genetic programming: a preliminary report. In *Proceedings of the Conference Companion on Genetic and Evolutionary Computation* (2012), pp. 401–408.
- [203] SPECTOR, L., LA CAVA, W., SHANABROOK, S., HELMUTH, T., AND PANTRIDGE, E. Relaxations of lexicase parent selection. In *Genetic Programming Theory and Practice XV*. 2018, pp. 105–120.
- [204] STECKE, K. E. Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems. *Management science* 29, 3 (1983), 273–288.
- [205] TAN, C. J., NEOH, S. C., LIM, C. P., HANOUN, S., WONG, W. P., LOO, C. K., ZHANG, L., AND NAHAVANDI, S. Application of an evolutionary algorithm-based ensemble model to job-shop scheduling. *Journal of Intelligent Manufacturing* 30, 2 (2019), 879–890.
- [206] TAY, J. C., AND HO, N. B. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering* 54, 3 (2008), 453–473.

- [207] TEYMOURIFAR, A., OZTURK, G., OZTURK, Z. K., AND BAHADIR, O. Extracting new dispatching rules for multi-objective dynamic flexible job shop scheduling with limited buffer spaces. *Cognitive Computation* 12, 1 (2020), 195–205.
- [208] TROISE, S. A., AND HELMUTH, T. Lexicase selection with weighted shuffle. In *Genetic Programming Theory and Practice XV*. 2018, pp. 89–104.
- [209] UMIĆ, M., AND JAKOBOVIĆ, D. Ensembles of priority rules for resource constrained project scheduling problem. *Applied Soft Computing* 110 (2021), 107606.
- [210] URASEVIĆ, M., GALA, F. J. G., JAKOBOVIĆ, D., AND COELLO, C. A. C. Combining single objective dispatching rules into multi-objective ensembles for the dynamic unrelated machines environment. *Swarm and Evolutionary Computation* (2023), 101318.
- [211] URASEVIĆ, M., AND JAKOBOVIĆ, D. Comparison of ensemble learning methods for creating ensembles of dispatching rules for the unrelated machines environment. *Genetic Programming and Evolvable Machines* 19 (2018), 53–92.
- [212] URASEVIĆ, M., AND JAKOBOVIĆ, D. Creating dispatching rules by simple ensemble combination. *Journal of Heuristics* 25 (2019), 959–1013.
- [213] URASEVIĆ, M., PLANINIĆ, L., GALA, F. J. G., AND JAKOBOVIĆ, D. Novel ensemble collaboration method for dynamic scheduling problems. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2022), pp. 893–901.
- [214] UY, N. Q., HOAI, N. X., O’NEILL, M., MCKAY, R. I., AND GALVÁN-LÓPEZ, E. Semantically-based crossover in genetic pro-

- gramming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines* 12, 2 (2011), 91–119.
- [215] UY, N. Q., MCKAY, B., O'NEILL, M., AND HOAI, N. X. Self-adapting semantic sensitivities for semantic similarity based crossover. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2010), pp. 1–7.
- [216] VAN HASSELT, H., GUEZ, A., AND SILVER, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2016), vol. 30. DOI:[10.1609/aaai.v30i1.10295](https://doi.org/10.1609/aaai.v30i1.10295).
- [217] VANNESCHI, L., CASTELLI, M., AND SILVA, S. A survey of semantic methods in genetic programming. *Genetic Programming and Evolvable Machines* 15 (2014), 195–214.
- [218] VEPSALAINEN, A. P., AND MORTON, T. E. Priority rules for job shops with weighted tardiness costs. *Management Science* 33, 8 (1987), 1035–1047.
- [219] VILCOT, G., AND BILLAUT, J. C. A tabu search algorithm for solving a multicriteria flexible job shop scheduling problem. *International Journal of Production Research* 49, 23 (2011), 6963–6980.
- [220] WANG, L., CAI, J., LI, M., AND LIU, Z. Flexible job shop scheduling problem using an improved ant colony optimization. *Scientific Programming* 2017 (2017), 1–11. DOI:[10.1155/2017/9016303](https://doi.org/10.1155/2017/9016303).
- [221] WANG, S., AND LI, Y. Variable neighbourhood search and mathematical programming for just-in-time job-shop scheduling problem. *Mathematical Problems in Engineering* 2014 (2014).
- [222] WANG, S., MEI, Y., AND ZHANG, M. Novel ensemble genetic programming hyper-heuristics for uncertain capacitated arc routing

- problem. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2019), pp. 1093–1101.
- [223] WANG, S., AND ZHANG, H. A matheuristic for flowshop scheduling with batch processing machines in textile manufacturing. *Applied Soft Computing* 145 (2023), 110594.
- [224] WANG, Z., LIANG, Z., ZENG, R., YUAN, H., AND SRINIVASAN, R. S. Identifying the optimal heterogeneous ensemble learning model for building energy prediction using the exhaustive search method. *Energy and Buildings* 281 (2023), 112763.
- [225] WEI, L., HE, J., GUO, Z., AND HU, Z. A multi-objective migrating birds optimization algorithm based on game theory for dynamic flexible job shop scheduling problem. *Expert Systems with Applications* 227 (2023), 120268.
- [226] WILCOXON, F., KATTI, S., AND WILCOX, R. A. Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test. *Selected Tables in Mathematical Statistics 1* (1970), 171–259.
- [227] WU, Z., FAN, H., SUN, Y., AND PENG, M. Efficient multi-objective optimization on dynamic flexible job shop scheduling using deep reinforcement learning approach. *Processes* 11, 7 (2023), 2018.
- [228] XIE, H. *An analysis of selection in genetic programming*. PhD thesis, Open Access Te Herenga Waka-Victoria University of Wellington, 2009.
- [229] XIE, H., AND ZHANG, M. Parent selection pressure auto-tuning for tournament selection in genetic programming. *IEEE Transactions on Evolutionary Computation* 17, 1 (2012), 1–19.

- [230] XIE, H., ZHANG, M., AND ANDREAE, P. Automatic selection pressure control in genetic programming. In *Proceedings of the International Conference on Intelligent Systems Design and Applications* (2006), vol. 1, pp. 435–440.
- [231] XIE, H., ZHANG, M., AND ANDREAE, P. An analysis of constructive crossover and selection pressure in genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2007), pp. 1739–1748.
- [232] XIE, H., ZHANG, M., ANDREAE, P., AND JOHNSON, M. An analysis of multi-sampled issue and no-replacement tournament selection. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation* (2008), pp. 1323–1330.
- [233] XIE, J., GAO, L., PENG, K., LI, X., AND LI, H. Review on flexible job shop scheduling. *IET Collaborative Intelligent Manufacturing* 1, 3 (2019), 67–77.
- [234] XIONG, H., FAN, H., JIANG, G., AND LI, G. A simulation-based study of dispatching rules in a dynamic job shop scheduling problem with batch release and extended technical precedence constraints. *European Journal of Operational Research* 257, 1 (2017), 13–24.
- [235] XIXING, L., AND YI, L. Approach of solving dual resource constrained multi-objective flexible job shop scheduling problem based on moea/d. *International Journal of Online Engineering* 14, 7 (2018), 75–89.
- [236] XU, B., MEI, Y., WANG, Y., JI, Z., AND ZHANG, M. Genetic programming with delayed routing for multiobjective dynamic flexible job shop scheduling. *Evolutionary Computation* 29, 1 (2021), 75–105.

- [237] XU, B., TAO, L., DENG, X., AND LI, W. An evolved dispatching rule based scheduling approach for solving djss problem. In *Proceedings of the Chinese Control Conference* (2021), pp. 6524–6531.
- [238] XU, M., MEI, Y., ZHANG, F., AND ZHANG, M. Genetic programming with cluster selection for dynamic flexible job shop scheduling. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2022), pp. 1–8. DOI:[10.1109/CEC55065.2022.9870431](https://doi.org/10.1109/CEC55065.2022.9870431).
- [239] XU, M., MEI, Y., ZHANG, F., AND ZHANG, M. Genetic programming and reinforcement learning on learning heuristics for dynamic scheduling: A preliminary comparison. *IEEE Computational Intelligence Magazine* (2023). DOI:[10.1109/MCI.2024.3363970](https://doi.org/10.1109/MCI.2024.3363970).
- [240] XU, M., MEI, Y., ZHANG, F., AND ZHANG, M. Genetic programming with lexicase selection for large-scale dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation* (2023). DOI:[10.1109/TEVC.2023.3244607](https://doi.org/10.1109/TEVC.2023.3244607).
- [241] XU, M., MEI, Y., ZHANG, F., AND ZHANG, M. Multi-objective genetic programming based on decomposition on evolving scheduling heuristics for dynamic scheduling. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2023), pp. 427–430.
- [242] YANG, W., LI, W., CAO, Y., LUO, Y., AND HE, L. Real-time production and logistics self-adaption scheduling based on information entropy theory. *Sensors* 20, 16 (2020), 4507.
- [243] YIN, W., LIU, M., AND WU, C. Learning single-machine scheduling heuristics subject to machine breakdowns with genetic programming. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2003), vol. 2, pp. 1050–1055.
- [244] YSKA, D., MEI, Y., AND ZHANG, M. Feature construction in genetic programming hyper-heuristic for dynamic flexible job shop

- scheduling. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (2018), pp. 149–150.
- [245] YSKA, D., MEI, Y., AND ZHANG, M. Genetic programming hyper-heuristic with cooperative coevolution for dynamic flexible job shop scheduling. In *Proceedings of the European Conference on Genetic Programming* (2018), pp. 306–321.
- [246] ZAKARIA, Y., ZAKARIA, Y., BAHAAELDIN, A., AND HADHOUD, M. Niching-based feature selection with multi-tree genetic programming for dynamic flexible job shop scheduling. In *Proceedings of the International Joint Conference on Computational Intelligence* (2021), pp. 3–27.
- [247] ZEITRÄG, Y., FIGUEIRA, J. R., HORTA, N., AND NEVES, R. Surrogate-assisted automatic evolving of dispatching rules for multi-objective dynamic job shop scheduling using genetic programming. *Expert Systems with Applications* 209 (2022), 118194.
- [248] ZEITRÄG, Y., RUI FIGUEIRA, J., AND FIGUEIRA, G. A cooperative coevolutionary hyper-heuristic approach to solve lot-sizing and job shop scheduling problems using genetic programming. *International Journal of Production Research* (2024), 1–28.
- [249] ZENG, Y., LIAO, Z., DAI, Y., WANG, R., LI, X., AND YUAN, B. Hybrid intelligence for dynamic job-shop scheduling with deep reinforcement learning and attention mechanism. *arXiv preprint arXiv:2201.00548* (2022).
- [250] ZHANG, C., SONG, W., CAO, Z., ZHANG, J., TAN, P. S., AND CHI, X. Learning to dispatch for job shop scheduling via deep reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 1621–1632.

- [251] ZHANG, C., YAO, X., TAN, W., ZHANG, Y., AND ZHANG, F. Proactive scheduling for job-shop based on abnormal event monitoring of workpieces and remaining useful life prediction of tools in wisdom manufacturing workshop. *Sensors* 19, 23 (2019), 5254.
- [252] ZHANG, F., MEI, Y., NGUYEN, S., TAN, K. C., AND ZHANG, M. Multitask genetic programming-based generative hyperheuristics: A case study in dynamic scheduling. *IEEE Transactions on Cybernetics* 52, 10 (2021), 10515–10528.
- [253] ZHANG, F., MEI, Y., NGUYEN, S., TAN, K. C., AND ZHANG, M. Instance rotation based surrogate in genetic programming with brood recombination for dynamic job shop scheduling. *IEEE Transactions on Evolutionary Computation* 27, 5 (2023), 1192–1206.
- [254] ZHANG, F., MEI, Y., NGUYEN, S., TAN, K. C., AND ZHANG, M. Task relatedness based multitask genetic programming for dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation* 27, 6 (2023), 1705–1719.
- [255] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. *IEEE Transactions on Cybernetics* 51, 4 (2020), 1797–1811.
- [256] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Genetic programming with adaptive search based on the frequency of features for dynamic flexible job shop scheduling. In *Proceedings of the European Conference on Evolutionary Computation in Combinatorial Optimization* (2020), pp. 214–230.
- [257] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Guided subtree selection for genetic operators in genetic programming for dynamic

- flexible job shop scheduling. In *Proceedings of the European Conference on Genetic Programming* (2020), pp. 262–278.
- [258] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Correlation coefficient based recombinative guidance for genetic programming hyper-heuristics in dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation* 25, 3 (2021), 552–566.
- [259] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Collaborative multifidelity-based surrogate models for genetic programming in dynamic flexible job shop scheduling. *IEEE Transactions on Cybernetics* 52, 8 (2022), 8142–8156.
- [260] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Phenotype based surrogate-assisted multi-objective genetic programming with brood recombination for dynamic flexible job shop scheduling. In *Proceedings of the IEEE Symposium Series on Computational Intelligence* (2022), pp. 1218–1225.
- [261] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Multi-task multi-objective genetic programming for automated scheduling heuristic learning in dynamic flexible job shop scheduling. *IEEE Transactions on Cybernetics* 53, 7 (2023), 4473–4486.
- [262] ZHANG, F., MEI, Y., NGUYEN, S., AND ZHANG, M. Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling. *IEEE Transactions on Evolutionary Computation* 28, 1 (2024), 147–167.
- [263] ZHANG, F., MEI, Y., NGUYEN, S., ZHANG, M., AND TAN, K. C. Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation* 25, 4 (2021), 651–665.

- [264] ZHANG, F., MEI, Y., NGUYEN, S., ZHANG, M., AND TAN, K. C. Surrogate-assisted evolutionary multitasking genetic programming for dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation* 25, 4 (aug 2021), 651–665.
- [265] ZHANG, F., MEI, Y., AND ZHANG, M. Genetic programming with multi-tree representation for dynamic flexible job shop scheduling. In *Proceedings of the Australasian Joint Conference on Artificial Intelligence* (2018), pp. 472–484.
- [266] ZHANG, F., MEI, Y., AND ZHANG, M. Surrogate-assisted genetic programming for dynamic flexible job shop scheduling. In *Proceedings of the Australasian Joint Conference on Artificial Intelligence* (2018), pp. 766–772.
- [267] ZHANG, F., MEI, Y., AND ZHANG, M. Evolving dispatching rules for multi-objective dynamic flexible job shop scheduling via genetic programming hyper-heuristics. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2019), pp. 1366–1373.
- [268] ZHANG, F., MEI, Y., AND ZHANG, M. A new representation in genetic programming for evolving dispatching rules for dynamic flexible job shop scheduling. In *Proceedings of the European Conference on Evolutionary Computation in Combinatorial Optimisation* (2019), pp. 33–49.
- [269] ZHANG, F., MEI, Y., AND ZHANG, M. A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2019), pp. 347–355.
- [270] ZHANG, F., MEI, Y., AND ZHANG, M. An investigation of terminal settings on multitask multi-objective dynamic flexible job shop

- scheduling with genetic programming. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (2023), pp. 259–262.
- [271] ZHANG, F., SHI, G., AND MEI, Y. Interpretability-aware multi-objective genetic programming for scheduling heuristics learning in dynamic flexible job shop scheduling. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2023), pp. 1–8. DOI:[10.1109/CEC53210.2023.10253970](https://doi.org/10.1109/CEC53210.2023.10253970).
- [272] ZHANG, H., CHEN, Q., TONDA, A., XUE, B., BANZHAF, W., AND ZHANG, M. Map-elites with cosine-similarity for evolutionary ensemble learning. In *Proceedings of the European Conference on Genetic Programming* (2023), pp. 84–100.
- [273] ZHANG, H., ZHOU, A., CHEN, Q., XUE, B., AND ZHANG, M. Sr-forest: A genetic programming based heterogeneous ensemble learning method. *IEEE Transactions on Evolutionary Computation* (2023). DOI:[10.1109/TEVC.2023.3243172](https://doi.org/10.1109/TEVC.2023.3243172).
- [274] ZHANG, J., DING, G., ZOU, Y., QIN, S., AND FU, J. Review of job shop scheduling research and its new perspectives under industry 4.0. *Journal of Intelligent Manufacturing* 30, 4 (2019), 1809–1830.
- [275] ZHANG, L., FENG, Y., XIAO, Q., XU, Y., LI, D., YANG, D., AND YANG, Z. Deep reinforcement learning for dynamic flexible job shop scheduling problem considering variable processing times. *Journal of Manufacturing Systems* 71 (2023), 257–273.
- [276] ZHANG, Q., AND LI, H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation* 11, 6 (2007), 712–731.

- [277] ZHANG, R., SONG, S., AND WU, C. A hybrid artificial bee colony algorithm for the job shop scheduling problem. *International Journal of Production Economics* 141, 1 (2013), 167–178.
- [278] ZHANG, Y., ZHU, H., TANG, D., ZHOU, T., AND GUI, Y. Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems. *Robotics and Computer Integrated Manufacturing* 78 (2022), 102412.
- [279] ZHAO, F., CHEN, Z., WANG, J., AND ZHANG, C. An improved moea/d for multi-objective job shop scheduling problem. *International Journal of Computer Integrated Manufacturing* 30, 6 (2017), 616–640.
- [280] ZHOU, B., AND WEN, M. A dynamic material distribution scheduling of automotive assembly line considering material-handling errors. *Engineering Computations* 40, 5 (2023), 1101–1127.
- [281] ZHOU, M., CHIU, H. S., AND XIONG, H. H. Petri net scheduling of fms using branch and bound method. In *Proceedings of IEEE Conference on Industrial Electronics* (1995), vol. 1, pp. 211–216.
- [282] ZHOU, Y., AND YANG, J. Automatic design of scheduling policies for dynamic flexible job shop scheduling by multi-objective genetic programming based hyper-heuristic. *Procedia CIRP* 79 (2019), 439–444.
- [283] ZHOU, Y., YANG, J., AND HUANG, Z. Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming. *International Journal of Production Research* 58, 9 (2020), 2561–2580.
- [284] ZHOU, Y., YANG, J., AND ZHENG, L. Hyper-heuristic coevolution of machine assignment and job sequencing rules for multi-objective dynamic flexible job shop scheduling. *IEEE Access* 7 (2018), 68–88.

- [285] ZHOU, Y., YANG, J., AND ZHENG, L. Multi-agent based hyper-heuristics for multi-objective flexible job shop scheduling: A case study in an aero-engine blade manufacturing plant. *IEEE Access* 7 (2019), 21147–21176.
- [286] ZHU, X., WANG, W., GUO, X., AND SHI, L. A genetic programming-based evolutionary approach for flexible job shop scheduling with multiple process plans. In *Proceedings of the International Conference on Automation Science and Engineering* (2020), pp. 49–54.
- [287] ZIMMERMAN, D. W., AND ZUMBO, B. D. Relative power of the wilcoxon test, the friedman test, and repeated-measures anova on ranks. *Journal of Experimental Education* 62, 1 (1993), 75–86.
- [288] ZITZLER, E., DEB, K., AND THIELE, L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation* 8, 2 (2000), 173–195.
- [289] ZITZLER, E., LAUMANN, M., AND THIELE, L. Spea2: Improving the strength pareto evolutionary algorithm. *TIK report* 103 (2001).
- [290] ZITZLER, E., THIELE, L., LAUMANN, M., FONSECA, C. M., AND DA FONSECA, V. G. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation* 7, 2 (2003), 117–132.