Genetic Programming for Vehicle Subset Selection in Ambulance Dispatching

Jordan MacLachlan, Yi Mei, Fangfang Zhang[✉], Mengjie Zhang
School of Engineering and Computer Science, Victoria University of Wellington, New Zealand
{jordan.maclachlan, yi.mei, fangfang.zhang, mengjie.zhang}@ecs.vuw.ac.nz

***Abstract*—Assigning ambulances to emergencies in real-time, ensuring both that patients receive adequate care and that the fleet remains capable of responding to any potential new emergency, is a critical component of any ambulance service. Thus far, most techniques to manage this problem are as convoluted as the problem itself. As such, many real-world medical services resort to using the naive *closest-idle* rule, whereby the nearest available vehicles are dispatched to serve each new call. This paper explores the feasibility of using a genetic programming hyper heuristic (GPHH) in order to generate intelligible rules of thumb to select which vehicles should attend any given emergency. Such rules, either manually or automatically designed, are evaluated within a novel solution construction procedure which constructs solutions to the ambulance dispatching problem given the parameters of the simulation environment. Experimental results suggest that GPHH is a promising technique to use when approaching the ambulance dispatching problem. Further, a GPHH-evolved rule's interpretability allows for detailed semantic analysis into which features of the environment are valuable to the decision making process, allowing for human dispatching agents to make more informed decisions in practice.**

***Index Terms*—Hyper Heuristic, Genetic Programming, Ambulance Dispatch, Evolutionary Computation**

I. Introduction

Ambulance dispatching is a critical challenge faced by many modern municipalities. Typically, policies in practice dispatch multiple crews to a single job so excess paramedics can perform auxiliary functions such as communicating with the public, drawing medication, advising leading staff or rotating through CPR. In the event of a medical emergency, two decisions must be made to ensure both that the fleet is appropriately deployed and that the patient receives sufficient care in a reasonable time. First, the call is triaged: given information provided from witnesses on site, a medically trained call-taker assigns the call an urgency code (e.g. purple, red, orange, green, etc.). Second, given the emergency's location and the newly acquired urgency code, a dispatcher selects and directs a number of vehicles to the call. The latter is the decision making process of interest in this work.

Current representations of emergency medical services (EMS) problems are diverse. Broadly speaking, EMS transportation research can be categorised into one of three problems: a) set covering, b) vehicle allocation, or c) vehicle dispatching (or more broadly, subset selection). Covering problems aim to identify the location of facilities a priori, such as to minimise the response time to any possible emergency [1]. Vehicle allocation problems aim to allocate a fleet of vehicles to a geographically set number of facilities such as to best handle an incoming call load [2] in an a priori and/or real-time manner. It is not uncommon to see these problems solved sequentially [3]. Ambulance dispatching, however, must be solved in response to emergencies as they occur, which is a hard optimisation problem.

Consider a graph of arc-connected nodes. A subset of nodes are designated *hospitals*, to which a fraction of patients must be transferred. Likewise a subset of nodes are *facilities*, at which a limited number of idle ambulances are permitted to wait. The objective of ambulance dispatching is to identify a subset of idle vehicles such that given a new arc-based emergency, a) the emergency demand requirements are met in minimal time, b) the first response time is minimised, and c) the first response time to the *next* (unknown) emergency is minimised. Several constraints must be adhered to during this process. For example, at least one ambulance of each required *type* must be present for its component of the total required treatment duration prior to the emergency being deemed complete. More ambulances of a type (up to the maximum) will speed up this process.

Often, in practice, selecting the closest idle vehicle is deemed an acceptable dispatching strategy. Sending the closest idle vehicle is easy to implement, understand, and audit, so it remains popular. However, it has been long-known that the *closest-idle* method is not always the most reliable technique [4], [5]. Somewhat counter-intuitively, instead, we must consider foregoing an optimal arrival time to the current patient in order to possibly serve the non-existent next patient, better. This principle is clearly evident in air rescue services, where often such resources are extremely limited and only deployed as a last resort [6], increasing the odds the equipment is available to those that absolutely need it.

While filtering calls into their respective urgency codes has an extensive foundation in medical and paramedic research [7], [8], vehicle subset selection does not. A "multiple-response" strategy was proposed in [9], allowing several vehicles to attend each event. Their work focuses on identifying which *type* of vehicle(s) should attend an emergency then identifying the specific response vehicles. They integrate into the reward function both set coverage theory and a function indicating the vehicle type's utility relative to task urgency. Such complex methods have proven effective to date and have therefore been extended, as in [10], to further iterate on the method.

However, the "closest-idle" heuristic is successful because

2022 IEEE Congress on Evolutionary Computation (CEC) | 978-1-6654-6708-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/CEC55065.2022.9870323

of its simplicity which is a characteristic current alternatives lack. Various manually designed heuristics have been proposed to overcome the bias toward short-term thinking of the closest-idle policy [11], [12]. Such heuristics aim not to always generate an optimal solution, but instead to quickly provide a high quality one. However, manually-designed heuristics are costly to develop as they rely heavily on the domain knowledge of experts. While these particular designs may be fast to deploy and provide a "good enough" solution in many cases, we suspect better, more generalisable rules exist.

The Genetic programming hyper heuristic (GPHH) algorithm has proven effective at exploring the heuristic space of complex problems. GPHH utilises Darwinian theory to combine a primitive set of functions and state variables to form program trees that are used in a simulation to construct solutions. In both scheduling and routing, GPHH learns heuristics that repeatedly compete at state-of-the-art level [13]–[15]. GPHH evolves an effective decision-making heuristic [16] employed within a domain-dependent solution construction procedure (SCP) [17]. The SCP of this work is an ambulance dispatching simulation environment. Despite its potential applicability to this problem domain, to the best of our knowledge, GPHH has not been applied to ambulance dispatching.

The primary goal of this paper is to perform an exploratory analysis of whether GPHH is fit for ambulance dispatching. There are several sub-objectives associated with this goal:

1) Formulate a baseline mathematical representation of the real-world problem faced by ambulance personnel.
2) Design a solution construction procedure to simulate ambulance dispatching in line with the new problem model.
3) Implement a pilot GPHH, including the design of several novel terminal features.
4) Examine the effectiveness of the new SCP and GPHH in comparison to existing manually designed rules.

We define the new problem in Section II-A before examining the existing literature in Section II-B and providing an overview of the general GPHH algorithm in Section III-A. Section III-B outlines the SCP and the implementation-specific aspects of the GPHH. We present our experiments and perform high-level analysis in Section V before closing with our intended research direction.

II. Background

A. Problem Definition: Ambulance Dispatching Problem

An ambulance dispatching problem (ADP) instance occurs on a directed graph $G = (V, A)$ consisting of vertices V and arcs (directed edges) A. Let $F \subset V$ be the set of homogeneous facilities, each with a maximum vehicle capacity Q. Let $H \subset V$ be the set of homogeneous hospitals, to which the necessary patients be transferred. Hospitals may double as facilities, in which case $|F \cap H| > 0$. Each arc $a \in A$ has a positive traversal *time* t_a.

A fleet of respondent vehicles R attend to calls. Three distinct subsets $R_1, R_2, R_3 \subset R$, respectively distinguish Advanced, Basic, and Community Life Support vehicles, denoted *ALS*, *BLS*, and *CLS*. Both ALS and BLS vehicles can transport one patient to hospital while CLS vehicles have zero capacity.

Each emergency call $i \in C$ has several characteristics:

- an arrival time m_i in minutes since "midnight";
- an arc on which the call has occurred $a_i \in A$;
- a point $af_i \in [0, 1]$ along a_i denoting precisely where the emergency occurred (in lieu of a street address);
- a demand vector $\mathbf{d}_i$ of non-negative integers, indicating the number and types of vehicles required in order to provide adequate care: $\mathbf{d}_i = [d_i^1, d_i^2, d_i^3]$;
- a positive treatment time τ_i required per unit of demand, thus total required treatment time is $\tau_i \mathbf{d}_i$; and,
- a binary variable $h_i \in \{0, 1\}$ indicating whether a patient requires hospitalisation.

A solution to an ADP instance is represented by a binary matrix $\mathbf{X}$ where $x_{ij} = 1$ indicates that vehicle j is assigned to call i and $x_{ij} = 0$ otherwise. The start and completion time of each vehicle for each call can be calculated from a simulation with non-delay strategy. The objective of ADP is to minimise the mean response time (r_i) plus true treatment time $(\hat{\tau}_i)$ over all calls. $\hat{\tau}_i$ is defined as the time between the first vehicle arriving on scene and when the patient is either discharged or en route to hospital. The objective is defined by Equation 1.

$$\min \ \frac{1}{|C|} \sum_{i=1}^{|C|} (r_i + \hat{\tau}_i). \tag{1}$$

We enforce several constraints to ensure a solution is valid:

- A vehicle can be assigned to an emergency only after the call is received.
- A vehicle can arrive at an emergency only after the time it is assigned.
- A vehicle can only be assigned to a single emergency at a time.
- An emergency must receive the required treatment time τ_i from each requested vehicle.
- No excess vehicles may be sent to an emergency, however fewer than the requested number may attend.
- Once an ambulance arrives on site one must remain until treatment is complete.
- All values of $\mathbf{X}$ must lie in the binary domain.

The above model is designed for the *ambulance* dispatching problem, however at its core, this task is simply a *Subset Selection Problem*. Remaining in an emergency context, we may similarly apply such a model to police or fire dispatching. In principle, our goal is to design a GPHH capable of selecting *several* members of a larger group.

B. Related Work

1) Ambulance Dispatching: we break the ambulance dispatching literature into two: those whose goal is to generate a solution to a specific instance and those whose goal is to generate a method that in itself can generate a solution to any instance.

Several evolutionary methods have been employed to solve instances directly, in particular PSO [18], ant colony optimisation [19], [20], and genetic algorithms [21]. Principles such as the Petal Algorithm [22]–[24] have been important in the development of this space [18]. Often, such problem considerations include stochastic travel times, considering real-world aspects such as traffic and speed limit restrictions. Stochastic programming is often applied to this area [9], [10], [25], [26], usually optimising both vehicle allocation in a first stage, then dispatching in a second. The authors of [9] introduce the concept of *multiple response*, whereby the model can send several vehicles to a single emergency. This is a very important real-world consideration as often having several paramedic staff on hand is critical to managing and obtaining all relevant information from witnesses on-site. In [10], the same authors additionally consider non-transport vehicles which are often used to handle non-urgent calls that do not require hospitalisation.

We have found three heuristics specific to ambulance dispatch. First, choose the closest-idle ambulance, however serve patients in severity order. Further, an ambulance en-route or present at a non-urgent call may be redirected should another vehicle be capable of replacing it within a given response time threshold [12]. Second, a region (a discretised "demand" node) is said to be more *prepared* if more vehicles are nearby. A more prepared region is able to contribute more vehicles to a new emergency [11]. Third, select the ambulance that can respond within a given response time threshold while also minimising marginal coverage; a function of both emergency demand and the availability quotients of neighbouring vehicles [5].

2) Genetic Programming Hyper Heuristic: The principle of the GPHH method is to use GP to evolve low-level heuristics [16], [27] that in turn construct solutions to a given problem. i.e. GPHH searches the space of heuristics.

GPHH has been extensively applied to transportation problems. In arc routing, heuristics are used as routing policies, determining an idle vehicle's next task [17], [28]. GPHH for arc routing has been used alongside other learning strategies such as ensemble learning [29], niching [14], and transfer learning [30], [31]. GPHH is also extremely effective in job shop scheduling both in its standard [32], [33] and multi-task representations [34]. Likewise, GPHH for scheduling has been paired with other learning techniques such as surrogate-assisted cooperative co-evolution [35].

There are many other fields in which GPHH has proven effective. Project scheduling [36], bi-level cloud pricing optimization [37], flow-shop scheduling [38], and online resource allocation [39] offer but a few examples. The intent here is to highlight the flexibility of GPHH. Such a breadth of success in tasks not dissimilar to ambulance dispatching award us some confidence that GPHH will adapt well to this problem.

Algorithm 1 The general framework of the GPHH

```
1: Randomly initialise a population of heuristics;
2: while the stopping criterion is not met do
3:     Evaluate the current population via a suitable SCP;
4:     Generate a child population via genetic operators;
5:     Select individuals from both the parent and child
       populations to form the next generation.
```

TABLE I
THE NEWLY PROPOSED TERMINAL UNITS FOR THE AMBULANCE DISPATCHING SCP.

Event	**Context**
$ArrivalRate$	The global rate at which calls have been arriving thus far.
$ArrivalTime$	The time (minutes since midnight) at which this call arrived.
$CurrentTime$	The current time (minutes since midnight).
$DistClosestAlt$	Time the closest other vehicle (of the candidate's type) is from the emergency.
$ElapsedTime$	The minutes since this call arrived.
$EnRoute$	A binary output indicating whether the candidate is currently assigned and travelling to a different emergency.
$HospitalReq$	A binary value to indicate whether this call requires hospitalisation.
$AssignedDef$	The number of vehicles of the candidate's type yet to be *assigned* to this call.
$Demand$	The number of vehicles of this vehicle's type required to attend this call.
$FracBusy$	The percentage of vehicles of this type that are currently performing a task.
$PresentDef$	The number of vehicles of the candidate's type yet to arrive at this call.
$QueueClearance$	The number of vehicles of the candidate's type required to clear the queue.
$TimeDistance$	The time required for the candidate vehicle to travel to this emergency.
$TimeRemaining$	Given the vehicles currently present, how much longer until the task is complete?

III. Proposed GPHH for Ambulance Dispatching

A. A New Genetic Programming Hyper Heuristic

Algorithm 1 shows an overview of the GPHH process [17], [28], [40]. In the context of ambulance dispatching, Algorithm 1 evolves dispatching policies; heuristics in the form of priority trees. A dispatching policy is used to evaluate the state, given a new call, and identify the precise vehicle(s) of each type that should respond. To evaluate the effectiveness of each dispatch policy, each is used to generate solutions to a set of training instances that change each generation. The fitness of the policy is set as the mean fitness value across its training set. Subtree crossover, subtree mutation, and reproduction constitute the genetic operators and the stopping criterion is reaching the maximal number of generations.

1) Genetic Programming Primitive Set: in order for a GPHH to successfully learn a domain, it must be provided with building blocks containing sufficient state information (terminals) from which it can evaluate the current environment. Our terminal set is described by Table I.

Functions operate on terminals. The function set used in this work consists of $\{+, -, *, \%, min, max, if\}$, where $\%$

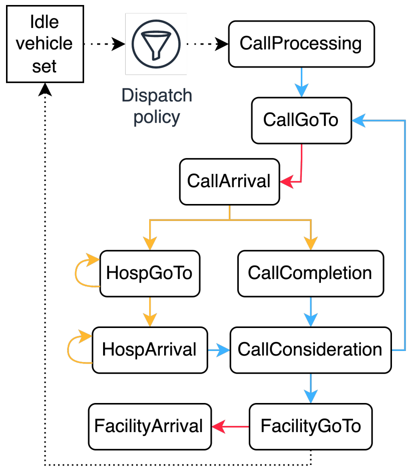

Fig. 1. The events system framework of the Solution Construction Procedure for Ambulance Dispatching. With relation to event *triggering* time, orange arrows indicate transitions that involve positive processing time; red, otherwise orange transitions that may be interrupted in order to serve a new, more urgent call (e.g. a vehicle en route to call A being redirected to call B); and blue, transitions that occur instantaneously. Black dotted lines show the passing of vehicles to and from the idle vehicle set.

represents protected division ($x/0 = 1$), and if returns the output of subtree 1 if subtree 0 is greater than 0 and subtree 2 otherwise. $\{+, -, *, min, max\}$ take their traditional arithmetic meanings. Together, the terminal and function sets constitute the *primitive* set.

B. A New Solution Construction Procedure

An SCP is a simulated invocation of a specified problem. In this case, that defined in Section II-A. While an SCP may abstract away from a real-world problem, ensuring an approximate representation is critical to the feasibility of applying a learned policy in practice.

1) Event Management for Ambulance Dispatching: Figure 1 shows a high-level overview of the SCP for ambulance dispatching. The SCP is a simulation where each discrete vehicle constructs a chain of temporally linear sequential *events*. Table II provides additional context for each of these events. Algorithm 2 shows the procedural steps by which the execution of these events is handled, which follows closely that of [17], [28], [40]. The pseudocode for a $CallProcessing$ event is shown in Algorithm 3 as an example of when and how the GPHH-evolved dispatch policies are used to make decisions.

IV. Experiment Design

In order to verify the effectiveness of the proposed GPHH method, we use two existing manually designed rules from the literature. First, the standard closest-idle rule (CIDP). This will allow us to compare our performance against many existing ambulance dispatching services internationally. Second, we borrow the concept of preparedness [11] and modify it to suit our particular use case. Instead of identifying the preparedness of each *facility* as in the original work, we calculate each

TABLE II
The Event Set Created During the Solution Construction Procedure for Ambulance Dispatching.

Event	Context
$CallProcessing$	Created upon receipt of a new call. A dispatch policy selects the vehicle(s) that will attend, creating one $CallGoTo$ for each.
$CallGoTo$	Created when a vehicle has been assigned to a call. Creates a $CallArrival$ to trigger at a calculated arrival time. May be interrupted by a $CallProcessing$ event.
$CallArrival$	Triggers when a vehicle arrives at an emergency. Calculates an estimated finish time, updates other vehicles assigned to this call (i.e. with the new completion time; en route vehicles may not need come) and creates either a $HospGoTo$ or $CallCompletion$ depending on C_h, this vehicle's type, and whether there are already vehicles present.
$HospGoTo$	Created when a vehicle has been designated as the transfer vehicle, or when waiting for another ambulance type to complete its service allocation. Executes on call completion, creating a $HospArrival$.
$HospArrival$	Triggered when a vehicle arrives at a hospital, creating another $HospArrival$ to trigger after 20 mins (a default value to account for transferring the patient Duty of Care to hospital staff).
$CallCompletion$	Triggered when a vehicle not assigned to transfer the patient has completed its component of the task. Creates a $CallConsideration$ event.
$CallConsideration$	Created when a vehicle actively becomes idle. It considers serving all emergencies that have already arrived but do not currently have enough vehicles assigned. Creates either a $FacilityGoTo$ or a $CallGoTo$ event.
$FacilityGoTo$	Created when there are no suitable emergencies for a newly idle vehicle to attend. Creates a $FacilityArrival$ event to trigger at the expected arrival time. May be interrupted by a $CallProcessing$ event.
$FacilityArrival$	Updates the vehicle's location and increments the facility's current vehicle load.

Algorithm 2 The solution construction procedure for ambulance dispatching.

```
Input: A problem instance $I$ (with $n$ calls), the ambulance fleet $R$, and
    dispatch policy $h(.)$.
Output: A solution $\mathbf{X} = (n, m)$
1: Initiliase an empty event queue $\Gamma$ and simulation state $\xi$
2: for each vehicle $r \in R$ do
3:     $\Gamma \leftarrow$ new $FacilityGoTo$ event at time 0
4: while $\Gamma$ is not empty do
5:     $\epsilon \leftarrow$ top (min start time) element removed from $\Gamma$
6:     trigger $\epsilon$ to update $\mathbf{X}$ and $\xi$
7: return $\mathbf{X} = (n, m)$
```

candidate vehicle's "preparedness": the fraction of the graph that it *uniquely* covers within a given response time threshold (*RTT* = 6 minutes). We denote this policy the Preparedness Dispatch Policy (PDP). The calculation of PDP is shown in Algorithm 4.

There is one distinct graph in Arc Routing's *EGL-G* dataset

Algorithm 3 Triggering a $CallProcessing$ event.

```
Input: the emergency $e$, state $\xi$, dispatch policy $h(.)$, and current time $\omega$.
 1: $R_i \leftarrow$ the idle vehicle set from $\xi$
 2: $R_f \leftarrow$ filter infeasible candidates from $R_i$
 3: if $R_f$ is empty then return $null$
 4: $iter \leftarrow$ empty priority queue
 5: for each candidate $r$ in $R_f$ do
 6:     $p_r \leftarrow h(r, e)$                                ▷ Apply the dispatch policy
 7:     add [$r$, $p_r$] to $iter$
 8: while $iter$ is not empty do
 9:     $top \leftarrow$ highest rated element of $iter$
10:     $veh \leftarrow top$[0]
11:     $p_{veh,e} \leftarrow top$[1]
12:     if $e$ has enough of $veh$ type vehicles then
13:         $continue$
14:     if $veh$ is en route to a call then ▷ Should we interrupt this vehicle?
15:         $miniPool \leftarrow$ a new candidate set containing only $veh$
16:         Update $\xi$, removing $curr$ as $veh$'s current task
17:         $p_{veh,curr} \leftarrow h(veh, curr)$              ▷ Apply the dispatch policy
18:         if $p_{veh,e} < p_{veh,curr}$ then                  ▷ Yes, interrupt
19:             Remove $veh$'s previous event from the event queue $\Gamma$
20:             $\Gamma \leftarrow$ new $CallGoTo$($\omega$, $veh$, $e$)
21:             Update $\xi$, adding $e$ as $veh$'s current task
22:         else                                                ▷ No, don't interrupt
23:             Update $\xi$, returning $curr$ as $veh$'s current task
24:     else if $veh$ at or en route to a facility then
25:         Update $\xi$, adding $e$ as $veh$'s current task
26:         Update $\xi$, leaving facility
27:         $\Gamma \leftarrow$ new $CallGoTo$($\omega$, $veh$, $e$)
28:     else
29:         Update $\xi$, adding $e$ as $veh$'s current task
30:         $\Gamma \leftarrow$ new $CallGoTo$($\omega$, $veh$, $e$)
31:     if $e$'s demand is not satisfied then
32:         $\varphi \leftarrow e$                         ▷ Add $e$ to the waiting call queue.
```

Algorithm 4 The calculation of the Preparedness policy.

```
Input: the current vehicle $veh$ and the set of vehicles $R$.
 1: $\Lambda \leftarrow$ new map from each vehicle to an empty set of arcs
 2: $\lambda \leftarrow$ new empty set of previously 'covered' tasks
 3: for each Vehicle $j \in R$ do
 4:     for each Arc $a \in A$ do
 5:         if dist($j$, $a$) $< RTT$ then
 6:             if $a$ not in $\lambda$ then
 7:                 add [$j$, $a$] to $\Lambda$
 8:                 add $a$ to $\lambda$
 9:             else
10:                 for each Vehicle $j \in R$ do
11:                     remove [$j$, $a$] from $\Lambda$
12: return $|\Lambda[veh]|$ / $|A|$
```

from which we generate seven unique graphs by first removing edges with zero-demand (providing *EGL-G1* and *EGL-G2*) then randomly removing an additional 5%. Note that we select the largest sub-graph should the resultant graph become disconnected. Upon each graph, *Instances* are generated, each representing one "day" in the life of an ambulance service. An instance consists of a random number of calls throughout the day given a temporally variant call frequency distribution (as shown in Figure 2), a bounded random number of A/B/CLS vehicles, and a random placing of a fixed number of hospitals and facilities.

At each generation during training, a rule generated by GPHH is evaluated on five different instances (i.e. five days). This five instance set is changed at each generation in order to force the population to continue learning. With 51 generations,

TABLE III
THE GENETIC PROGRAMMING TRAINING PARAMETERS.

Parameter	**Value**	**Parameter**	**Value**
Population size	1024	Crossover rate	80%
Generations	51	Mutation rate	15%
Tournament size	7	Reproduction rate	5%
Maximal depth	8		

TABLE IV
THE TEST PERFORMANCE OF THE CIDP, PDP, AND AVERAGE GPDP DISPATCH POLICIES ON THE *EGL-G* DATASET.

			Test Fitness		
Instance	$\lvert V\rvert$	$\lvert A\rvert$	CIDP	PDP	GPDP
EGL-G1-1	244	329	67.58	64.67	62.54(0.29)
EGL-G1-2	248	334	68.17	65.29	63.10(0.24)
EGL-G1-3	249	336	77.76	74.21	71.17(0.33)
EGL-G2-1	255	358	36.88	36.08	35.99(0.01)
EGL-G2-2	254	366	36.80	36.00	35.91(0.01)
EGL-G2-3	253	354	36.97	36.13	36.05(0.01)
EGL-G2-4	255	362	36.90	36.08	35.99(0.01)
Overall mean	251.14	348.43	53.99	49.78	48.68(-)
EGL-G1 mean	247.00	333.00	71.07	68.06	65.60(0.29)
EGL-G2 mean	254.25	360.00	36.91	36.07	35.99(0.01)

this means there are a total of 255 instances in the '*training set*'. During testing, the best rule of each generation is applied to 500 instances in the '*test set*', whereby the average total cost defines the policy's final effectiveness. Their average fitness determines the fitness of an algorithm on that graphical base instance (i.e. the original *EGL-GX-Y* instance). We repeat such tests using 30 unique random seeds in order to identify statistical significance via Wilcoxon Rank Sum.

The SCP and GPHH algorithms are implemented in Java using the ECJ Library [43]. The remaining genetic programming parameters are highlighted in Table III, following settings used previously in the Uncertain Arc Routing literature.

V. RESULTS AND DISCUSSIONS

Table IV shows the performance of the three different dispatch policies on the seven tested base instances. Note that only the new method, GPDP, shows its standard deviation as this is a measure of how variant a number of policies act across the test set. CIDP and PDP are single policy methods so therefore cannot be assigned such a measure. Figure 3 shows the distribution of GPDP test fitness on a single instance, in this case *EGL-G1-1*, in comparison to the fitness values of CIDP and PDP.

We make several observations of Table IV:

- Overall, performance significantly decreases on *EGL-G1*-based instances over *EGL-G2*-based instances (i.e. the total-cost is higher). This is most likely because critical edges in the *EGL-G1* instance files contained zero demand and were therefore excluded from the graph prior to the random 5% arc exclusion. Figure 4 shows the *EGL-G* base graphs (i.e. the distinct graphs after removing zero-demand edges, prior

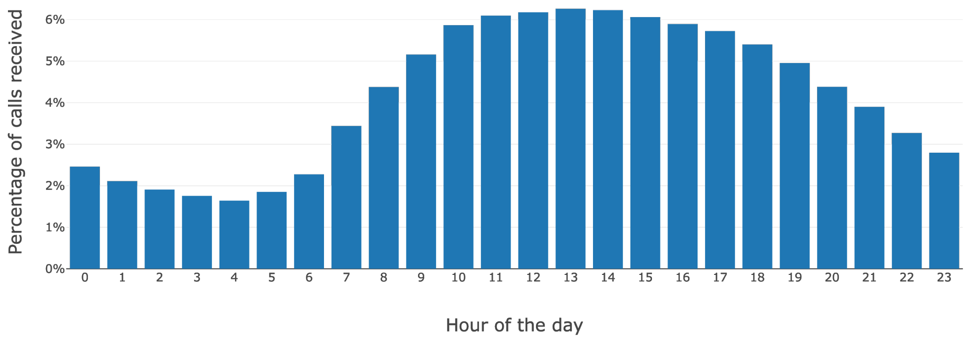

Fig. 2. Call arrival variance in the real world. This trend is observed in the Marin County emergency dataset [41] and in several other works [9], [42].

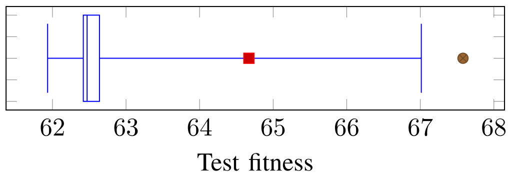

Fig. 3. The distribution of $GPDP$ test fitness on *EGL-G1-1*. The red square represents the PDP test fitness; brown, $CIDP$.

to random edge removal), in which it is evident that critical edges are missing from the *EGL-G1* graph, particularly through the central column (highlighted in yellow). As a result, traversing from one side of the graph to the other would become more limited and therefore likely more expensive.

- PDP outperforms CIDP on all instances. This advantage is particularly evident on the *EGL-G1*-based instances (a mean improvement of 4.33% vs 2.22%). We therefore deduce that the PDP policy is better able to handle more sparse networks by identifying the vehicles that offer unique coverage to hard to access areas of the graph.
- Despite the GPDP being a preliminary implementation, on all instances, the proposed method outperforms both manual comparison rules.
- For a raw GPHH-based method, GPDP is particularly stable on the *EGL-G2*-based instances. Such behaviour suggests that GPHH repeatedly finds a local optima and cannot escape. Examining several of the best performing *EGL-G2-4* test rules would suggest this is not necessarily the case. Figures 5 and 6 show two such rules:
  - The first rule strongly prefers vehicles that are near to the emergency, however additionally selects a vehicle more frequently if its type is of high demand in the queue or the arrival rate is high. Given the current SCP only considers queued emergencies when a vehicle *becomes idle*, this rule may have learned to select vehicles of the needed types in order to add them to the emergency queue pipeline.
  - The second rule balances the ratio between the closest alternative vehicle to the emergency and the candidate vehicle, preferring vehicles that have an alternative nearby.

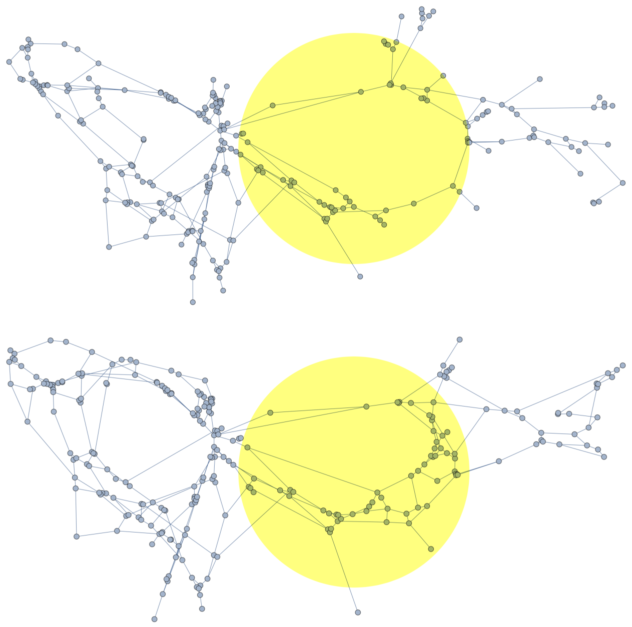

Fig. 4. All EGL-G1 (G-2) graphs are constructed from the upper (bottom) graph, with 5% of edges randomly removed.

    Given the requirement of multiple response for many tasks, the rule may have learned to avoid sending a vehicle to a task at which they simply have to wait (due to the no gap Constraint).

  Both rules are fundamentally different in their approach to selecting vehicles, yet achieve a very similar test fitness. Perhaps instead of being caught in the same local minima, GP has reached its performance limit with the primitive set provided, and there are multiple paths to the same objective. Deeper analysis is required in order to understand the true decisions being made by each rule to better understand where differences lie.

The size of rules generated by GP is of particular interest to scholars, as a smaller rule is typically faster to compute and easier for a human to parse. Figure 7 shows the average program size across all instances. This correlates with the improvement in test performance observed, shown in Figure

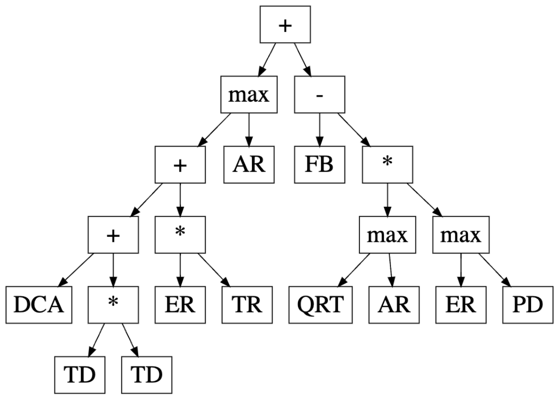

Fig. 5. An example of an effective rule learned on Instance *EGL-G2-4*.

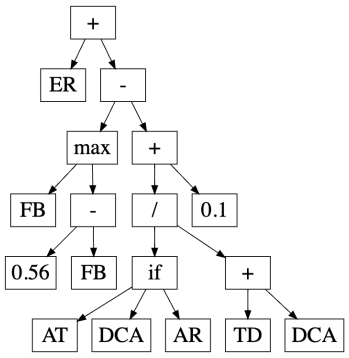

Fig. 6. An example of an effective rule learned on Instance *EGL-G2-4*.

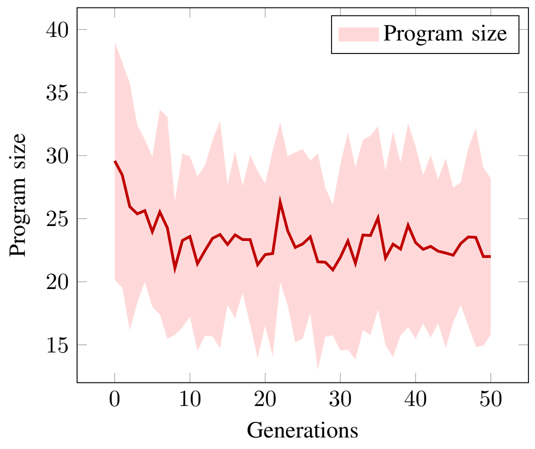

Fig. 7. The average evolved program size.

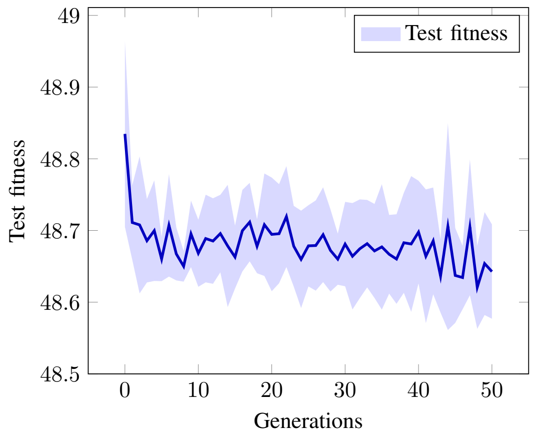

Fig. 8. The average $GPDP$ test fitness.

8. Interestingly, without direct instruction to do so, the GPHH has learned that smaller rules are more effective. This directly tracks with one of our primary motivations for using GP: to develop more interpretable dispatch policies.

VI. Conclusions and Future Work

The primary goal of this work was to determine whether the GPHH method was capable of learning dispatch policies for Ambulance Dispatching. Specifically, could GPHH design a policy that, given an unexpected emergency and a set of idle ambulances, could select a better suited subset of vehicles than the intuitive and commonly used CIDP. Without a doubt, we have shown that GPHH can outperform such naïve dispatch methods. Having defined a mathematical model for the Ambulance Dispatching Problem, we design a novel SCP that adheres to the problem constraints in which we can evaluate the effectiveness of any given dispatch policy. The proposed SCP provides a core foundation from which future works can build and continue to evaluate methods. A preliminary GPHH model is proposed and assessed on a series of instances generated from the well-known *EGL-G* Arc Routing dataset against two manually designed rules, including the closest-idle rule. GPHH is shown to perform best on all tested instances, marking promise for the technique in the field.

There are several areas we intend to explore in future works. First, we need to better understand several discrepancies of this paper: a) why there is such a fitness variation between *EGL-G1* and *EGL-G2*-based instances, b) why GPDP has such a low standard deviation on *EGL-G2*-based instances, and c) to gain a better understanding of how the learned dispatch policies behave in and impact the solutions generated by the SCP. Second, we intend to expand the problem model to step the representation further towards reality. To do so, we shall begin to include factors such as uncertain traversal times, duty of care transfers, and signal, flexible ambulance demand, and crew shift window limitations.

Finally, if you are the author of an Ambulance Dispatching or Subset Selection method and are willing to share your source code, we would like to use your work as a benchmark. Please contact us via the emails above.

References

[1] C. Toregas, R. Swain, C. ReVelle, and L. Bergman, "The location of emergency service facilities," *Operations research*, vol. 19, no. 6, pp. 1363–1373, 1971.

[2] M. S. Maxwell, S. G. Henderson, and H. Topaloglu, "Tuning approximate dynamic programming policies for ambulance redeployment via direct search," *Stochastic Systems*, vol. 3, no. 2, pp. 322–361, 2013.

[3] P. Beraldi and M. E. Bruni, "A probabilistic model applied to emergency service vehicle location," *European Journal of Operational Research*, vol. 196, no. 1, pp. 323–331, 2009.

[4] G. M. Carter, J. M. Chaiken, and E. Ignall, "Response areas for two emergency units," *Operations Research*, vol. 20, no. 3, pp. 571–594, 1972.

[5] C. J. Jagtenberg, S. Bhulai, and R. D. van der Mei, "Dynamic ambulance dispatching: is the closest-idle policy always optimal?" *Health Care Management Science*, vol. 20, no. 4, pp. 517–531, Dec. 2017.

[6] P. Collins, C. Ellis, H. MacLeod, D. Ohs, T. Smith, A. Stevenson, and A. Swain, "Wellington Free Ambulance Clinical Procedures and Guidelines, Comprehensive Edition (2019 - 2022)," Wellington Free Ambulance, Tech. Rep., 2019.

[7] S. Thakore, E. McGugan, and W. Morrison, "Emergency ambulance dispatch: is there a case for triage?" *Journal of the Royal Society of Medicine*, vol. 95, no. 3, pp. 126–129, 2002.

[8] S. Postma, J.-H. E. Dambrink, M.-J. de Boer, A. M. Gosselink, G. J. Eggink, H. van de Wetering, F. Hollak, J. P. Ottervanger, J. C. Hoorntje, E. Kolkman *et al.*, "Prehospital triage in the ambulance reduces infarct size and improves clinical outcome," *American Heart Journal*, vol. 161, no. 2, pp. 276–282, 2011.

[9] S. Yoon and L. A. Albert, "A dynamic ambulance routing model with multiple response," *Transportation Research Part E: Logistics and Transportation Review*, vol. 133, p. 101807, 2020.

[10] S. Yoon, L. A. Albert, and V. M. White, "A stochastic programming approach for locating and dispatching two types of ambulances," *Transportation Science*, vol. 55, no. 2, pp. 275–296, Mar. 2021.

[11] T. Andersson, S. Petersson, and P. Varbrand, "Calculating the preparedness for an efficient ambulance health care," in *Proceedings of the IEEE Conference on Intelligent Transportation Systems*. IEEE, 2004, pp. 785–790.

[12] M. Gendreau, G. Laporte, and F. Semet, "A dynamic model and parallel tabu search heuristic for real-time ambulance relocation," *Parallel computing*, vol. 27, no. 12, pp. 1641–1653, 2001.

[13] F. Zhang, Y. Mei, and M. Zhang, "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*. IEEE, 2019, pp. 1366–1373.

[14] S. Wang, Y. Mei, M. Zhang, and X. Yao, "Genetic programming with niching for uncertain capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 1, pp. 73–87, 2022.

[15] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Multitask genetic programming-based generative hyperheuristics: A case study in dynamic scheduling," *IEEE Transactions on Cybernetics*, 2021.

[16] N. Pillay and R. Qu, *Hyper-heuristics: theory and applications*. Springer, 2018.

[17] J. MacLachlan, Y. Mei, J. Branke, and M. Zhang, "Genetic programming hyper-heuristics with vehicle collaboration for uncertain capacitated arc routing problems," *Evolutionary Computation*, vol. 28, no. 4, pp. 563–493, 2020.

[18] T. Tlili, M. Harzi, and S. Krichen, "Swarm-based approach for solving the ambulance routing problem," *Procedia Computer Science*, vol. 112, pp. 350–357, 2017.

[19] E. Mouhcine, Y. Karouani, K. Mansouri, and Y. Mohamed, "Toward a distributed strategy for emergency ambulance routing problem," in *Proceedings of the International Conference on Optimization and Applications*. IEEE, 2018, pp. 1–4.

[20] H. Wang, R. Xu, X. Zijie, X. Zhou, Q. Wang, Q. Duan, and X. Bu, "Research on the optimized dispatch and transportation scheme for emergency logistics," *Procedia Computer Science*, vol. 129, pp. 208–214, Jan. 2018.

[21] J. Cao, H. Han, Y. P. Jiang, and Y. J. Wang, "Emergency rescue vehicle dispatch planning using a hybrid algorithm," *International Journal of Information Technology & Decision Making*, vol. 17, no. 06, pp. 1865–1890, 2018.

[22] A. Wren and A. Holliday, "Computer scheduling of vehicles from one or more depots to a number of delivery points," *Journal of the Operational Research Society*, vol. 23, no. 3, pp. 333–344, 1972.

[23] B. E. Gillett and L. R. Miller, "A heuristic algorithm for the vehicle-dispatch problem," *Operations Research*, vol. 22, no. 2, pp. 340–349, 1974.

[24] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet, "Classical and modern heuristics for the vehicle routing problem," *International Transactions in Operational Research*, vol. 7, no. 4-5, pp. 285–300, 2000.

[25] P. Beraldi, M. E. Bruni, and D. Conforti, "Designing robust emergency medical service via stochastic programming," *European Journal of Operational Research*, vol. 158, no. 1, pp. 183–193, 2004.

[26] S. Enayati, O. Y. Özaltın, M. E. Mayorga, and C. Saydam, "Ambulance redeployment and dispatching under uncertainty with personnel workload limitations," *IISE Transactions*, vol. 50, no. 9, pp. 777–788, 2018.

[27] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Correlation coefficient based recombinative guidance for genetic programming hyper-heuristics in dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 3, pp. 552–566, 2021.

[28] Y. Mei and M. Zhang, "Genetic programming hyper-heuristic for multi-vehicle uncertain capacitated arc routing problem," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2018, pp. 141–142.

[29] S. Wang, Y. Mei, and M. Zhang, "Novel ensemble genetic programming hyper-heuristics for uncertain capacitated arc routing problem," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2019, pp. 1093–1101.

[30] M. A. Ardeh, Y. Mei, and M. Zhang, "Genetic programming hyper-heuristic with knowledge transfer for uncertain capacitated arc routing problem," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2019, pp. 334–335.

[31] ——, "Genetic programming with knowledge transfer and guided search for uncertain capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, 2021, DOI: 10.1109/TEVC.2021.3129278.

[32] H. Fan, H. Xiong, and M. Goh, "Genetic programming-based hyper-heuristic approach for solving dynamic job shop scheduling problem with extended technical precedence constraints," *Computers & Operations Research*, p. 105401, 2021.

[33] F. Zhang, S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: An evolutionary learning approach," in *Machine Learning: Foundations, Methodologies, and Applications*. Springer, 2021. DOI: 10.1007/978-981-16-4859-5, pp. XXXIII+338.

[34] F. Zhang, Y. Mei, S. Nguyen, M. Zhang, and K. C. Tan, "Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 651–665, 2021, DOI: 10.1109/TEVC.2021.3065707.

[35] Y. Zhou, J.-j. Yang, and Z. Huang, "Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming," *International Journal of Production Research*, vol. 58, no. 9, pp. 2561–2580, 2020.

[36] J. Lin, L. Zhu, and K. Gao, "A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem," *Expert Systems with Applications*, vol. 140, p. 112915, 2020.

[37] E. Kieffer, G. Danoy, M. R. Brust, P. Bouvry, and A. Nagih, "Tackling large-scale and combinatorial bi-level problems with a genetic programming hyper-heuristic," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 44–56, 2019.

[38] H.-B. Song and J. Lin, "A genetic programming hyper-heuristic for the distributed assembly permutation flow-shop scheduling problem with sequence dependent setup times," *Swarm and Evolutionary Computation*, vol. 60, p. 100807, 2021.

[39] B. Tan, H. Ma, Y. Mei, and M. Zhang, "A cooperative coevolution genetic programming hyper-heuristic approach for on-line resource allocation in container-based clouds," *IEEE Transactions on Cloud Computing*, 2020.

[40] Y. Liu, Y. Mei, M. Zhang, and Z. Zhang, "Automated heuristic design using genetic programming hyper-heuristic for uncertain capacitated arc routing problem," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2017, pp. 290–297.

[41] C. County of Marin, "Emergency medical service (ems) incidents," Jan 2022. [Online]. Available: https://data.marincounty.org/Public-Health/Emergency-Medical-Service-EMS-Incidents/swth-izpe

[42] Bureau of Emergency Medical Services, Pennsylvania Department of Health, "2019 EMS data report," Oct. 2020.

[43] S. Luke *et al.*, "A java-based evolutionary computation research system," https://cs.gmu.edu/~eclab/projects/ecj/, 2016.