Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Solver-Informed Evolution of Interpretable Dispatching Rules for the Stochastic Team Orienteering Problem with Time Windows

Augusto Magalhães Pinto de Mendonça

augustompm@id.uff.br

Universidade Federal Fluminense

Filipe Pessôa Sousa

Universidade do Estado do Rio de Janeiro

Laura Silva de Assis

Federal Center for Technological Education Celso Suckow da Fonseca

Igor Machado Coelho

Universidade Federal Fluminense

Research Article

Keywords: genetic algorithms, genetic programming hyper-heuristics, dispatching rules, team orienteering, stochastic optimisation

Posted Date: September 7th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10744997/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Solver-Informed Evolution of Interpretable Dispatching Rules for the Stochastic Team Orienteering Problem with Time Windows

Augusto Mendonça[1*], Filipe Pessôa Sousa[2], Laura Silva de Assis[3], Igor Machado Coelho[1]

[1*]Instituto de Computação, Universidade Federal Fluminense, Niterói, RJ, Brazil.
[2]Universidade do Estado do Rio de Janeiro, Rio de Janeiro, RJ, Brazil.
[3]Federal Center of Technological Education of Rio de Janeiro, CEFET/RJ, Rio de Janeiro, RJ, Brazil.

*Corresponding author(s). E-mail(s): augustompm@id.uff.br;
Contributing authors: filipe.sousa@pos.ime.uerj.br;
laura.assis@cefet-rj.br; imcoelho@ic.uff.br;

Abstract

The stochastic team orienteering problem with time windows (STOPTW) requires a team of vehicles to collect prizes under service-time uncertainty and hard time windows, choosing the next customer online. Genetic programming (GP) hyper-heuristics evolve interpretable dispatching rules for this setting, but the terminal set, the features the rules compose from, has remained the eleven of the neighbourhood-score GP (NS-GP) baseline. We propose SI-GP, a solver-informed genetic programming hyper-heuristic that extracts heuristic building blocks from high-quality reference solutions and selects an instance-specific subset of these features. Routes from the reference solvers are read move by move to identify the operations that separate them from the baseline rules. The terminals implementing these operations are grounded in the scheduling and orienteering literature, and two novel composites put downstream loss and reachable gain in one number. A structured-population genetic algorithm performs the per-instance selection, evaluating every chromosome by a full GP hyper-heuristic run on a GPU. A cross-validatory choice keeps the baseline as a backup member. The deployed rule needs no solver and no simulation, and never underperforms the baseline on the held-out set. SI-GP improves the NS-GP baseline by 1.3% on average on the 30 of the 40 benchmark instances where the baseline does

not already serve every customer, with 10 gains surviving Holm correction. The evolved rules remain readable, and on most instances their run-to-run deviation falls below the baseline's. The pattern is consistent with the solver-informed vocabulary regularising the GP search.

Keywords: genetic algorithms, genetic programming hyper-heuristics, dispatching rules, team orienteering, stochastic optimisation

1 Introduction

Wildfires burned 3.9 million square kilometres of land in the 2023–2024 fire season alone (Jones et al. 2024). Unmanned aerial systems already support the detection, monitoring, and active response phases of these events (Haeri Boroujeni et al. 2024). When a small team of vehicles responds, more hotspots demand attention than the fleet can reach. Each hotspot carries a different value, stays treatable only inside a window before it spreads, and takes an uncertain time to suppress once a vehicle arrives. Not every hotspot can be visited. The decision that matters is which ones to serve and in which order, while the clock runs. The same structure governs skimmer vessels containing an oil slick before it drifts ashore, spraying drones treating crop infestations inside biological windows, crews reinforcing dikes ahead of a flood crest or treating mosquito breeding sites within the vector's reproductive cycle, and even the sequencing of orbital debris removal, where the cost of each transfer shifts with the departure epoch (Federici et al. 2019). Operations research abstracts this family as the team orienteering problem with time windows: a fleet collects heterogeneous prizes under hard time windows and a shared time budget, and leaving customers unserved is part of the model rather than a failure of it. The problem entered the evolutionary computation literature through personalised tourist trip design (Mei and Zhang 2018), and the benchmark used in this work descends from that line.

Service durations in these operations are not known in advance. How long a suppression run takes depends on wind and on fuel conditions, and a containment manoeuvre depends on the sea state. Mei and Zhang (2018) observed that under stochastic visit durations a preplanned trip may become infeasible, since a single delay can push later visits outside their windows. Jackson and Mei (2020) likewise note that solutions produced by traditional optimisation techniques are inflexible in the face of variance in the stochastic variables. The observation extends to the strongest planners available. We replayed the routes produced by a high-quality reference solver under stochastic service times, and the frozen plans lost to a simple adaptive rule in 10 of the 30 `Solomon` instances (Section 3.1). Unlike conventional offline routing, where a complete route is determined before service durations are observed, the policy considered here selects the next customer online after each realised service duration. A dispatching rule, also called a policy or priority function, ranks the feasible customers at every decision point and commits only to the next visit, absorbing each deviation as it happens. Genetic programming hyper-heuristics automatically evolve such rules (Burke et al. 2013; Nguyen et al. 2013). For the stochastic TOPTW, Mei and

Zhang (2018) introduced the GPHH that evolves priority functions over 10 scheduling features, and Jackson and Mei (2020) extended the terminal set with a neighbourhood score, establishing the NS-GP baseline used throughout this work.

The evolved rule is a compact artefact. It is an arithmetic expression over features of the current state, a few dozen nodes at most, and evaluating it for one candidate costs a fixed handful of operations. A decision therefore involves no search and no simulation, and no solver is called at any point. In our experiments, a complete held-out evaluation of thirty rules over five hundred scenarios runs in seconds, which puts a single dispatch decision in the microsecond range on commodity hardware. The training cost is paid once, before deployment. Burke et al. (2013) argue that generating a heuristic is expensive only in the short term, because one run of the evolution yields a method that answers future instances far more quickly than re-running an optimiser. Nguyen et al. (2013) make the same point when they contrast reusable rules with the disposable solutions of conventional metaheuristics. For the operations described above, this is the property that matters. A rule this small can be read, and it can also be embedded, so a vehicle carries its complete decision logic without any connection to a ground station.

The quality of an evolved rule is bounded by the vocabulary it composes from. Nguyen et al. (2013) found that evolved rules improve when system and machine attributes enter the terminal set, and the survey of Zhang et al. (2024) is explicit that terminals are not limited to the published tables and should be defined for the scenario under examination. The stochastic TOPTW lineage has nevertheless kept the same eleven features since the NS-GP baseline was established. In dynamic job shop scheduling, richer vocabularies have been pursued through feature construction and feature selection (Yska et al. 2018; Mei et al. 2017; Zhang et al. 2021), yet in every case the features are evolved from scratch or selected once for a whole problem class. None of these methods consults a solver. High-quality offline solvers for routing now exist and run on GPUs, and their routes encode operations that no baseline rule performs (Wouda et al. 2024). Reading those routes is not imitation learning. Ingimundardottir and Runarsson (2018) learn dispatching preferences by labelling states with optimal decisions and report degradation once the learned rule leaves the teacher's distribution. We take the opposite path and extract a vocabulary, letting the GP re-optimise every rule directly on the stochastic objective.

A single vocabulary is still a compromise, because instances of different regimes reward different features. The feature selection methods above choose one subset for an entire scenario (Zhang et al. 2021), and the recent instance-specific framework of Zhang et al. (2025) generates heuristics as opaque code through a large language model on deterministic problems. Selecting an interpretable subset for each instance through a wrapper was, until recently, computationally out of reach. Mei and Zhang (2018) report an average training time of 966 seconds for their basic GP and 26,182 seconds for the look-ahead variant (about seven hours). The difference comes from exactly the kind of neighbourhood feature our vocabulary multiplies. Our engine removes this barrier. A single CUDA kernel evaluates entire populations with the instance resident on the device, following the design line of recent GPU frameworks for evolutionary computation (Wu et al. 2026; Lan et al. 2022; Huang et al. 2025), and completes thirty

full GP runs in 36 to 1,092 seconds per instance. One complete GPHH execution becomes cheap enough to serve as a single fitness evaluation. A structured-population genetic algorithm (Toledo et al. 2014) can then search the space of terminal subsets per instance, spending a median of 113 full executions per instance and over 5,000 in the campaign reported here.

The main objective of this work is to evolve interpretable dispatching rules that capture part of the gap to high-quality offline solvers while deciding without any simulation at deployment. Three sub-objectives follow: (*i*) constructing a terminal vocabulary from the routes of the reference solvers and grounding it in the scheduling and orienteering literature, (*ii*) for each instance selecting the subset of that vocabulary the GP composes from, and (*iii*) validating the evolved rules under a paired protocol, explaining where and why they gain. The resulting method, SI-GP, reads the solver routes move by move, implements the observed operations as 18 terminals in four families, 16 grounded in published terminals and two novel composites. Then, it searches for the subsets per instance using a genetic algorithm with a structured population whose fitness function is a full GPHH execution. A cross-validatory choice with the baseline as a backup member closes the pipeline. On the 40 benchmark instances, SI-GP is never worse than the NS-GP baseline, with 36 wins, 4 draws and 10 gains surviving Holm correction at a global Wilcoxon $p = 1.7 \times 10^{-7}$. On the 30 instances where the baseline does not already serve every customer, the average improvement is 1.3%, and over all 40 it is 1.0%. In 29 of the 38 evolved instances, the deviation across runs falls below the baseline, and in the strongest cases the per-seed floor rises outright. The evidence points to the solver-informed vocabulary regularising the GP search.

Our contributions are fourfold. First, we construct a solver-informed terminal vocabulary whose provenance is traceable link by link, from route reading to literature grounding to two novel composite terminals. Second, we select terminal subsets per instance with a structured-population genetic algorithm acting as a true wrapper over the GPHH, an approach previously ruled out by the cost of each evaluation. Third, we validate the 40-instance campaign under common random numbers with paired Wilcoxon tests, Holm correction, and effect sizes, and we identify the mechanism behind the gains. Fourth, we release the engine, the instances, the complete campaign logs, and thirty-seed traces in a public repository[1].

According to our literature search, this is the first study to extract an interpretable terminal vocabulary from the routes generated by a high-quality reference solver and to perform instance-specific selection of these terminals within a genetic programming hyper-heuristic. Although related approaches have explored the joint generation and selection of heuristics for deterministic vehicle routing problems (VRPs), and more recent studies have investigated instance-specific heuristic generation using large language models, these approaches neither exploit solver-generated routes as a source of interpretable heuristic knowledge nor generate transparent rules specifically for stochastic routing problems. Our approach therefore establishes a distinct framework that combines solver-derived knowledge, instance-specific heuristic selection, and interpretable genetic programming for stochastic routing.

[1] https://github.com/augustompm/solver-informed-dispatching-rules.

The remainder of this paper is organised as follows. Section 2 reviews the problem and related work. Section 3 presents SI-GP, from the reference routes and the terminal construction to the per-instance selection. Section 4 details the experimental setup and reports the results, the statistical analysis, the interpretability findings and the computational cost. Concluding remarks, together with limitations and future work, are provided in Section 5.

2 Background and related work

2.1 The stochastic team orienteering problem with time windows

The Orienteering Problem (OP) was introduced by Tsiligirides (1984) in the context of score-orienteering events, where a competitor selects a subset of control points to visit so as to maximise the collected score subject to a prescribed time limit. Chao et al. (1996) subsequently extended the formulation to a team of M members, with each member following an individual route and the objective defined as the total score collected by the team. The incorporation of time-window constraints leads to the Team Orienteering Problem with Time Windows (TOPTW). The deterministic variants of this problem family have been extensively studied and are comprehensively reviewed by Vansteenwegen et al. (2011) and Gunawan et al. (2016). With respect to benchmark instances, two main sources have been widely adopted in the literature. The `Solomon` instances, originally introduced for the VRP with Time Windows (VRPTW), represent clustered, randomly distributed, and mixed customer configurations over both short and long scheduling horizons (Solomon 1987). The `pr` instances originate from the multi-depot vehicle routing problems of Cordeau et al. (1997) and were subsequently adapted to the TOPTW by Montemanni and Gambardella (2009). These instances contain up to 288 customers and are characterised by substantially wider time windows.

Mei and Zhang (2018) introduced the stochastic TOPTW considered in this study, in which service durations are modelled as random variables with a standard deviation equal to 20% of their mean in the benchmark instances. Consequently, the actual service duration is revealed only when a customer is being served. Stochastic extensions of the OP family, however, predate this formulation. Campbell et al. (2011) considered stochastic travel and service times in a single-vehicle setting, incorporating penalties for commitments that cannot be fulfilled. Jin and Thomas (2019) investigated a team-based variant in which both rewards and service durations are uncertain, motivated by intrahospital routing; however, the routes are determined a priori, before the realisation of uncertainty. More recently, simheuristic approaches have incorporated Monte Carlo simulation within metaheuristic frameworks and have been applied to several stochastic variants of the TOP (Panadero et al. 2022; Herrera et al. 2022; Panadero et al. 2024). Learning-based approaches have also been investigated, including reinforcement learning for orienteering with stochastic release dates (Li et al. 2024) and neural constructive methods under stochastic travel times (Hottung and Tierney 2022) or with service duration treated as a decision variable (Gao et al. 2026). Despite these advances, the literature that addresses such studies does not provide an

interpretable decision rule capable of making online routing decisions under stochastic service durations without relying on simulation at decision time. Existing approaches either determine routes in advance, explicitly sample scenarios during the decision process, or consider sources of uncertainty other than service duration, such as travel times or release dates. It is also worth noting that the acronym STOPTW has been used by Yu et al. (2024) to denote the Set Team Orienteering Problem with Time Windows, a deterministic variant in which customers are partitioned into clusters. In the present study, STOPTW denotes the stochastic Team Orienteering Problem with Time Windows, following the terminology introduced by Mei and Zhang (2018).

2.2 Genetic programming hyper-heuristics and dispatching rules

Hyper-heuristics search over heuristics (or over heuristic components) rather than directly over problem solutions (Burke et al. 2013). Two main families are distinguished: hyper-heuristics that select among existing heuristics and those that generate new ones (Burke et al. 2013; Drake et al. 2020). Genetic programming (Koza 1992) is the predominant framework for heuristic generation. For dispatching rules in job shop scheduling, Nguyen et al. (2013) compared three representations, ranging from arithmetic trees that produce composite rules to decision trees that select rules according to machine attributes. Regardless of representation, the expressive capacity of evolved rules is determined by the terminal set, which provides the scheduling statistics from which they are constructed. Grammar-based variants further constrain the search to syntactically valid structures. Grammatical evolution has been used to generate perturbative solvers (Sabar et al. 2013), local search heuristics (Burke et al. 2012), and constructive heuristics for the capacitated VRP (Marshall et al. 2014).

The stochastic TOPTW was introduced in the literature by Mei and Zhang (2018). Besides defining the problem, the authors incorporated two look-ahead features into the GPHH to anticipate the effect of a visit on later opportunities. Karunakaran et al. (2019) subsequently trained rules across instances using multitask GP, while Jackson and Mei (2020) added cluster information through a constructed terminal feature. The resulting NS-GP, built on the neighbourhood-score feature, is the reference dispatching rule for the problem. According to our literature search, these three studies constitute the complete GPHH literature on the stochastic TOPTW, with no subsequent contributions identified since 2020. Moreover, the most recent survey of the orienteering problem, covering the period from 2017 to 2025, does not include genetic programming or hyper-heuristics among the reviewed methodological families (Shen et al. 2025). This work revisits this research direction by addressing the question of which terminals should constitute the building blocks of the evolved rules.

2.3 Richer vocabularies: feature construction and selection in GPHH

A related line of research in dynamic job shop scheduling considers the vocabulary itself as an object to improve. Yska et al. (2018) co-evolved constructed features together with the rules, and identified feature selection over the terminal set as a natural

extension of this approach. Mei et al. (2017) addressed this selection problem with a niching-based framework that employs a surrogate model to reduce the computational cost of each fitness evaluation, thereby mitigating the expense of assessing feature importance through complete GP runs. Zhang et al. (2021) proposed a two-stage design in which the population is re-initialised using only the selected features, one subset per scheduling scenario. The survey by Zhang et al. (2024) places these methods within the broader adoption of machine learning techniques for heuristic design. Inside the STOPTW lineage, the cluster-awareness terminal introduced by Jackson and Mei (2020) further showed the potential contribution of a carefully designed constructed feature. As previously stated, our work departs from this line mainly in the source and selection of candidate terminals. Rather than evolving features from scratch, the candidate terminals implement operations read from the routes generated by a high-quality reference solver and are grounded in the established problem-specific literature. Moreover, the terminal subset is chosen per instance, not once per entire scenario. The selection procedure follows a true wrapper, *i.e.*, each candidate subset is evaluated by running the complete GPHH with the corresponding terminal set. This computationally demanding strategy becomes tractable through the accelerated evaluation engine described in Section 2.6.

2.4 Learning from high-quality solutions

A separate thread learns decision behaviour directly from high-quality or optimal solutions. Ingimundardottir and Runarsson (2018) framed job shop dispatching as a preference learning problem over post-decision states labelled by an exact solver. However, the resulting rules may become unreliable once their own choices lead to states outside the trajectories represented in the training data, a limitation also reported by the authors. Acero and Li (2024) distilled neural locomotion policies into gradient boosting models and then into symbolic expressions. Although the intermediate model preserves the expert policy, the final symbolic representation incurred a substantial loss in accuracy. Sharifi et al. (2025) derived driving policies from demonstrations through inductive logic programming. In scheduling, Wei et al. (2026) calibrated the selection among given classical dispatching rules using rollout-based labels and an uncertainty gate, while Ferreira et al. (2022) guided GP search with insights from problem-specific knowledge, reaching compact rules for dynamic job shops. In these approaches, the source primarily provides behavioural labels or guidance for selecting or ranking existing rules. In contrast, our approach uses the reference solver as a source of structural knowledge rather than as a target for imitation or rule selection. Specifically, the routes generated by the solver are analysed to identify relevant operations and, consequently, to construct candidate terminal features. The GP subsequently optimises these solver-informed building blocks directly with respect to the stochastic objective, without an imitation target or behavioural supervision at any stage.

2.5 Per-instance selection and portfolios

Choosing among algorithms on a per-instance basis is a well-established research problem. Rice (1976) formalised it as a mapping from instance features to the best-performing member of an algorithm space, using models such as weighted linear combinations or decision trees. Kotthoff (2014) observes that research in this area increasingly focuses on "identifying the most suitable existing algorithm" rather than developing new ones. Selection hyper-heuristics extend the same principle to heuristic search by choosing low-level heuristics during an iterative process (Drake et al. 2020). Recent instantiations still operate over predefined rules, either by learning to imitate expert selectors with neural networks (Tyasnurita et al. 2024) or by using metaheuristics to select a fixed set of priority rules (Garza-Santisteban et al. 2025). Algorithm configuration, *e.g.* with `irace`, answers the complementary question of tuning the parameters of a single algorithm (López-Ibáñez et al. 2016). In practice, `SATzilla` built per-instance solver portfolios using empirical hardness models and kept the overall best solver as a backup (Xu et al. 2008), and `ASlib` later standardised the evaluation of such systems (Bischl et al. 2016). Selection based on held-out performance also has a long history, dating back to cross-validatory model selection (Stone 1974). Our selection stage follows these principles, with the NS-GP baseline included as an always-available portfolio member.

Two systems are particularly close to our framework. Sim and Hart (2016) evolved constructive heuristics for the VRP using GP and subsequently applied a perturbative hyper-heuristic to select among the resulting candidates. This represents the closest structural predecessor to our work, although its generation stage is not guided by solver-derived knowledge and the problem is deterministic. Zhang et al. (2025) partitioned the instance space into subclasses, generated one heuristic per subclass with a large language model, and subsequently allowed the model to select among them online. In that framework, the resulting heuristics are executable code. Across this whole literature, existing portfolio approaches consider members that are predefined, imitated from experts, evolved without solver guidance, or generated by language models. In contrast, the vocabulary from which our portfolio members are constructed is informed by the routes produced by a high-quality reference solver.

2.6 Reference solvers and GPU acceleration

This work reads routes produced by high-quality solvers for the deterministic problem; two of them are part of the pipeline. `PyVRP` implements hybrid genetic search and ranked first in the 2021 DIMACS VRPTW challenge and in the static track of the EURO meets NeurIPS 2022 competition (Wouda et al. 2024). NVIDIA `cuOpt` is a GPU-accelerated routing engine that iteratively improves a population of candidate solutions[2]. Neither solver addresses the stochastic objective considered here. Both are used exclusively offline to obtain reference routes, and the per-instance reference gap is quantified in Section 3.

GPU acceleration has also been increasingly adopted in evolutionary computation. General-purpose metaheuristic frameworks can now run entirely on the device

[2] https://docs.nvidia.com/cuopt, accessed July 2026.

(Liu 2026), tensor based formulations accelerate VRP local search (Lei et al. 2025), and `EvoX` scales evolutionary pipelines across accelerators (Huang et al. 2025). For GP in particular, vectorised engines have reported speedups approaching three orders of magnitude (Baeta et al. 2021), while `EvoGP` exploits population-level parallelism for tree-based GP (Wu et al. 2026). On the reinforcement learning side, `WarpDrive` demonstrated the benefits of keeping the entire learning loop on a single GPU to avoid data transfer overhead (Lan et al. 2022). The computational challenge is particularly relevant to our problem lineage: the look-ahead variant introduced by Mei and Zhang (2018) multiplies training time by a factor of 27 relative to basic GP, while a solver-informed vocabulary adds several terminals of exactly that kind. Our engine follows this computational paradigm by keeping the instance resident on the device and evaluating rules bit-exactly against the CPU reference, thereby reducing a complete GPHH execution to an affordable evaluation unit. This makes the per-instance wrapper described in Section 3.4 computationally feasible.

2.7 Positioning our work in the literature

Table 1 positions SI-GP against the works reviewed in this section, following the assumption tables of Shen et al. (2025). The five attribute columns read as follows. Uncertain environment: data revealed during execution, stochastic or dynamic. Real-time decisions: the method decides while the plan executes. Interpretable rule: the deployed logic is a readable expression. Solver-derived knowledge: high-quality or optimal solutions enter the method. Instance-specific selection: the heuristic or vocabulary is chosen per instance. The orienteering family either fixes its routes in advance or learns policies that are not interpretable. The GPHH lineage of the stochastic TOPTW deploys interpretable rules that decide online, with a vocabulary that remained fixed from Mei and Zhang (2018) to Jackson and Mei (2020), and the vocabulary studies in scheduling construct or select features at most once per scenario. The works that learn from high-quality solutions transfer behavioural labels or rankings. The generation-with-selection line operates on deterministic problems and consults no solver. No other row fills all five columns. SI-GP does, and the knowledge it derives from the solver is structural: the routes inform which terminals to construct.

Taken together, these research directions define a clear methodological gap. First, GPHH vocabularies are typically evolved from scratch or selected once per scenario. Second, portfolio members are predefined, learned through imitation, or generated without reference to a solver. Third, information extracted from high-quality solutions is generally transferred as behavioural labels rather than as structural knowledge. Moreover, the GPHH literature on the stochastic TOPTW has remained inactive since 2020. Our literature search identified no prior study that analyses routes generated by a reference solver to determine which terminals should be available to a GPHH, nor one that performs this vocabulary selection on an instance-specific basis. The search was conducted using Scopus and Google Scholar in August 2026, with the exact queries, database exports, and screening records provided in the public repository. The next section describes the two components of this framework and their integration.

Table 1 Positioning of SI-GP against the works reviewed in this section, following the assumption tables of Shen et al. (2025). The attribute columns are defined in the text.

Reference	Problem	Algorithm	Uncertain environment	Real-time decisions	Interpretable rule	Solver-derived knowledge	Instance-specific selection
The orienteering family (Section 2.1)							
Campbell et al. (2011)	OPSTS	variable neighbourhood search, dynamic programming for special cases	✓	–	–	–	–
Jin and Thomas (2019)	TOP, stochastic rewards and service times	a priori approach with solution sampling	✓	–	–	–	–
Panadero et al. (2022); Herrera et al. (2022); Panadero et al. (2024)	stochastic TOP variants	simheuristics, biased-randomised algorithms	✓	–	–	–	–
Li et al. (2024)	OP, stochastic and dynamic release dates	reinforcement learning, value function approximation with exact ILPs	✓	✓	–	–	–
Hottung and Tierney (2022)	TD-OPSWTW	deep reinforcement learning	✓	✓	–	–	–
Gao et al. (2026)	OPTWVP	transformer with linear programming, 2 stages	–	–	–	–	–
GPHH for the stochastic TOPTW (Section 2.2)							
Mei and Zhang (2018)	STOPTW	GPHH	✓	✓	✓	–	–
Karunakaran et al. (2019)	STOPTW	multitasking GPHH	✓	✓	✓	–	–
Jackson and Mei (2020)	STOPTW	GPHH, constructed NS terminal	✓	✓	✓	–	–
Terminal vocabularies in GPHH (Section 2.3)							
Yska et al. (2018)	DFJSS	GPHH, feature construction	✓	✓	✓	–	–
Mei et al. (2017)	DJSS	GPHH, feature selection with a surrogate	✓	✓	✓	–	–
Zhang et al. (2021)	DFJSS	2-stage GPHH, feature selection per scenario	✓	✓	✓	–	–
Learning from high-quality solutions (Section 2.4)							
Ingimundardottir and Runarsson (2018)	JSS	preference learning with `DAgger`	–	–	✓	✓	–
Wei et al. (2026)	JSS	selection hyper-heuristic with an uncertainty gate	–	–	✓	✓	–
Ferreira et al. (2022)	DJSS	GP, guided empirical learning	✓	✓	✓	–	–
Per-instance selection and portfolios (Section 2.5)							
Sim and Hart (2016)	VRP	GP constructive heuristics, selective hyper-heuristic	–	–	✓	–	✓
Zhang et al. (2025)	OBPP, CVRP	LLM generation and selection	✓	✓	–	–	✓
Xu et al. (2008)	SAT	algorithm portfolio, empirical hardness models	–	–	–	–	✓
SI-GP (this work)	STOPTW	GPHH, GA wrapper, cross-validatory choice	✓	✓	✓	✓	✓

3 Solver-informed genetic programming

SI-GP is assembled in four design-time stages, summarised in Figure 1. Design time: solver execution, route analysis, terminal construction, terminal selection and GP evolution. Deployment time: evaluation of the evolved arithmetic rule only. Two reference solvers produce high-quality routes for the deterministic counterpart of each instance, which are subsequently replayed under stochastic realisations to quantify the prize left uncollected by the baseline. Analysing these routes move by move turns that gap into a list of recurring operations, implemented as 18 constructed terminals with explicitly identified origins. A structured-population genetic algorithm then selects, instance by instance, the subset of terminals the GP composes from, while the final stage performs a cross-validatory choice between the evolved rule and the baseline, which is retained as a backup portfolio member. The resulting deployed artefact remains an arithmetic expression over state features that makes decisions in microseconds without solver calls or simulation. The remainder of this section first describes the reference route and terminal vocabulary, followed by the selection procedure introduced in Section 3.4.

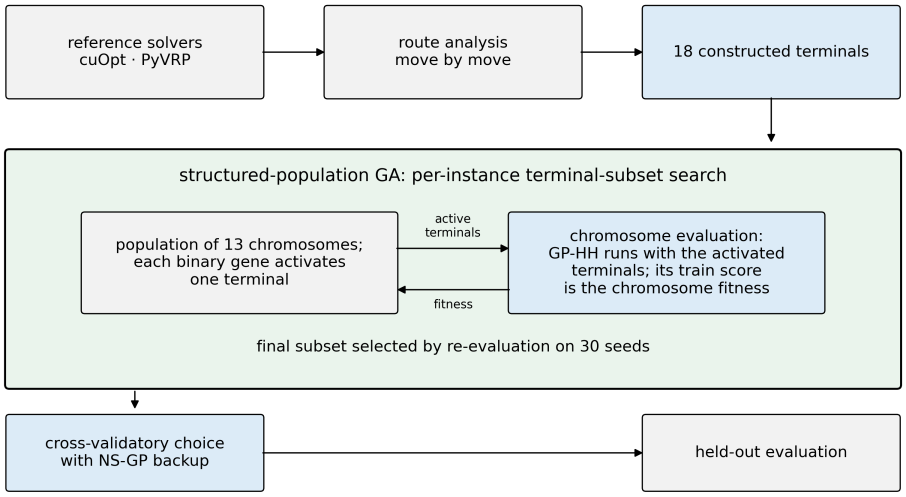


Fig. 1 The SI-GP pipeline. Every stage runs at design time, and the deployed rule decides without search, solver, or simulation.

3.1 Reference routes and the per-instance gap

We first solve each instance in its nominal form, with all service durations fixed at their expected value. NVIDIA `cuOpt` runs on the GPU under a time budget of 900 seconds per instance, while `PyVRP` repeatedly restarts its hybrid genetic search until no further improvement of the incumbent is obtained. The pair was chosen for benchmark fit and for independence: the instances descend from classical VRP benchmarks, `PyVRP`

ranked first in the two routing competitions cited in Section 2.6, and `cuOpt` runs a population-based search of a different design on the GPU. Taking the best of the two routes per instance keeps the reference ceiling from depending on a single search paradigm. Neither solver accounts for the stochastic objective, and the introduction already showed what happens when their routes are executed as fixed plans. Here, the routes are used for measurement instead. Each route is replayed over the 500 held-out scenarios of the evaluation protocol (Section 4.1), following the standard practice of scoring a fixed policy by sampled simulation (Tesauro and Galperin 1996; Secomandi 2001). For each instance, the reference gap is defined as the difference between the best mean performance obtained by the two reference routes and that of the NS-GP baseline. This replay happens at design time, on the same scenario set that later scores the evolved rules. Thus, the gap compares the baseline and reference routes under identical stochastic realisations, while the fixed routes cannot adapt to their outcomes.

Across the 40 instances, the gap averages 4.2% and its geography is sharp. Eight of the ten Cordeau instances exhibit gaps above 6%, reaching 11.5% for pr05. The random and mixed short-horizon `Solomon` groups yield gaps between 4.2% and 9.7%, whereas the clustered instances remain below 3.4%. The long-horizon series splits between moderate gaps (4.2% on rc201, 2.9% on r201) and saturated instances in which no additional reward remains to be collected. Overall, the routes outperform the baseline on 30 of the 40 instances and underperform it on the remaining ten, all of which are long-horizon or tightly clustered instances, where the baseline already serves almost every customer. The two independent solvers provide consistent evidence of this performance ceiling: they tie exactly on 22 of the 40 instances, the margin between them stays below one percent on all but two, and `PyVRP` produces the best route on five instances. Accordingly, the gap measures the problem rather than solver-specific behaviour.

A positive gap indicates the potential for an improved decision rule, but does not imply that the corresponding reference route should be deployed. For the c101, for example, the fixed reference route loses 0.7% to the adaptive baseline once under stochastic service durations, and the vocabulary derived from that same route still enables the evolved rule to outperform the baseline by 1.4% (Section 4). Thus, the improvement arises from extracting structural information from the reference route, not from imitating its decision. Figure 2 shows the c101 geometry, with the reference route and the two rules side by side. Each rule is shown at its best held-out seed, a retrospective pick for display only. The reference route reaches distant, narrow-window customers with less total travel, the pattern the reading turns into terminals.

3.2 Reading the routes

The reading is deliberately mechanical, and it is published in full. For each `Solomon` instance, two reading tools are available in the repository that compare the reference route with the thirty baseline rules replayed at $\sigma = 0$. The first tool reports coverage and losses. It lists customers served by the reference route but not by the rules, splits them into lost by all thirty seeds and lost by the median, profiles them by prize, window width, opening time, and distance from the depot, and locates them along the route timeline. The second tool measures four classical mechanics on the same routes

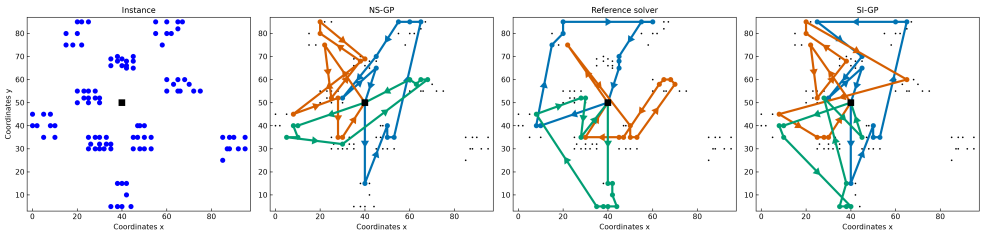


Fig. 2 c101, left to right: the instance, the NS-GP tour, the reference solver route, and the SI-GP tour, decoded at $\sigma = 0$ with three vehicles. The black square is the depot and the small dots are unserved customers.

before any terminal exists: visiting order against depot distance, waiting structure, angular sweep, and mean savings. Aggregated over the thirty instances, the reference routes exhibit greater angular sweep, 0.61 against 0.55 on the sector score, while the correlation between visiting order and depot distance is weakly but consistently negative, -0.12 against $+0.04$. Waiting depends on the horizon. In short-horizon groups, a reference plan waits about half as much as a rule plan, 23 against 43 time units, while in long-horizon groups it waits twice as much, 291 against 146, spending slack that saturated instances leave free. The rule-side statistics use the five representative seeds retained in each log.

The sixty logs, two per instance, reproduce byte for byte from the public repository.

The reading covers the thirty `Solomon` instances on which this work began. The ten Cordeau instances were introduced later, after the vocabulary had been established. The per-instance selection in Section 3.4 adapts the vocabulary to these instances, and Section 4.4 shows that they elect the same families.

c101 illustrates what the reading returns. The reference route serves 32 of the 100 customers against a median of 30 for the rules and collects 810 prize units versus a median of 780, while travelling less and waiting less: 513 distance units against 659, and 72 against 110 time units of waiting, respectively, relative to the median seed. One customer is lost by all thirty seeds. It pays 20 inside a window of 65 time units and sits 31 distance units from the depot, against an instance average of 28.9. Customers served by every seed exhibit the opposite profile, averaging 20.4 distance units from the depot with a prize of 30. In this case, the interpretation is direct: the rules favour nearby customers and readily collectible prize, whereas the reference route serves more distant, narrow-window customers at the cost of tighter regional tours.

Move by move, the operations that separate the solver from the rules can be reduced to four questions that a rule could ask before committing to a candidate: *(i)* how much prize stays reachable a few steps beyond it, *(ii)* what downstream costs the commitment induces, in terms of customers lost, prize lost, and lateness risk, *(iii)* how that cost compares with the value that remains reachable, and *(iv)* how much of the horizon is already gone. The eleven baseline features answer none of these questions directly. The look-ahead pair of Mei and Zhang (2018) came closest, at the training cost quantified in Section 2.6. The four terminal families in Table 2 implement exactly these four questions.

3.3 The solver-informed vocabulary

Constructing terminals for the scenario at hand follows the established practice in the field, as documented in the terminal tables reproduced by the survey (Zhang et al. 2024), and the baseline itself follows this approach, with the neighbourhood score introduced as a constructed feature with explicit motivation (Jackson and Mei 2020). We extend this practice at scale, with the source declared. The eleven baseline terminals are inherited unchanged. Sixteen of the eighteen constructed terminals are grounded in published terminals, while two are novel composites. All terminals are algebraic functions of the decision state, defined below and summarised in Table 2.

Throughout, p is the candidate under evaluation, q ranges over unvisited customers with prize $s(q)$, window $[o(q), c(q)]$ and expected duration $E[d(q)]$, $t(\cdot,\cdot)$ is travel time, and $T_{\max}$ the shared budget. The vehicle sits at position v at time τ, and $\hat{f}_p = \max(\tau + t(v,p),\, o(p)) + E[d(p)]$ is the expected finish of the candidate. Realised durations follow the benchmark model, normal with mean $E[d(q)]$ and standard deviation $\sigma E[d(q)]$, truncated at zero, with $\sigma = 0.2$. Every probability below is a closed-form tail of the untruncated Gaussian. The cut sits five standard deviations below the mean and leaves that approximation intact. The simulator clamps its draws at zero. A customer is feasible from a position and a clock when it can be reached inside its window and the vehicle can still return in time. Writing $a(q) = \tau + t(v,q)$ for the arrival,

$$F(v,\tau) = \big\{q \text{ unvisited} : a(q) \le c(q),\ \max(a(q),\, o(q)) + E[d(q)] + t(q,0) \le T_{\max}\big\}, \quad (1)$$

and $\mathrm{NFEAS} = |F(v,\tau)|$ is the queue-length analogue.

The reachability family asks what remains reachable beyond the candidate. Our bit-exact replication of the baseline computes the neighbourhood score as

$$\mathrm{NS}(p) = \sum_{q \in F(p,\, \hat{f}_p)} s(q)/t(p,q), \quad (2)$$

the prize rate of everything still open after p. MAXNS keeps the largest term of that sum instead. NSTW weighs each term by the unspent fraction of the window, $\max(0,\, 1 - w_q/(c(q) - o(q)))$, with w_q the waiting time the visit would impose. NSROB evaluates the same sum with the candidate finishing late by $\sigma E[d(p)]$ and every successor duration inflated to $(1+\sigma)E[d(q)]$. REACHk walks k greedy steps instead of looking one ahead,

$$\mathrm{REACH}k(p) = \sum_{i=1}^{k} s(q_i), \qquad q_i = \arg\max_{q \in F(q_{i-1},\,\cdot)} s(q)/t(q_{i-1},q), \qquad q_0 = p, \quad (3)$$

with times propagated by expected durations and $k \in \{1,2,3,5\}$, and REACH2ROB applies the NSROB stress to the two-step walk. The family generalises the one-step NPT-style look-ahead of the scheduling tables (Zhang et al. 2024) and the look-ahead pair of Mei and Zhang (2018).

The window-cascade family asks what committing to p destroys or endangers downstream. The customers lost by the commitment are

$$L(p) = F(v,\tau) \setminus F(p,\, \hat{f}_p), \qquad \text{NLOST} = |L(p)|, \qquad \text{SLOST} = \sum_{q \in L(p)} s(q). \tag{4}$$

REGRET entered the vocabulary independently, as a deterministic reading of the expected regret of Bian and Liu (2018), and turns out algebraically identical to SLOST: both measure the prize mass the commitment forfeits. The vocabulary kept both genes, the selection treats them as synonyms, and no winning subset elects the two together (Section 4.4). ATRISK is the chance-constrained counterpart of SLOST. Instead of counting a successor as kept or lost, it weighs every open successor by its in-time arrival probability (Varakantham et al. 2018; Bian and Liu 2018) under the uncertainty of the candidate's own duration,

$$\text{ATRISK}(p) = \sum_{q \in F(v,\tau)} s(q)\,\big(1 - \text{IAP}_q(p)\big), \tag{5}$$

where $\text{IAP}_q(p)$ multiplies two Gaussian factors, meeting the window of q and returning inside the budget, both driven by $\sigma E[d(p)]$. IAPD keeps only the sharpest of these risks,

$$\text{IAPD}(p) = \tfrac{1}{2}\,\text{erfc}\!\left(\frac{c(q^*) - \hat{a}(q^*)}{\sqrt{2}\,\sigma E[d(p)]}\right), \tag{6}$$

the probability that the window of the best downstream successor q^* is missed, with $\hat{a}(q^*)$ its expected arrival: one minus the in-time arrival probability of the successor the greedy walk would take next.

The two novel composites put both sides of a commitment into one number: LOSSR $=$ SLOST$/$(NS $+ 1$), the cascade damage per unit of reachable value, and NETLOSS $=$ SLOST $-$ NS, its net damage. Unlike the other sixteen constructed terminals, neither has a published parent: they are the only features here that weigh the cascade loss against the reachable value, and no terminal table we know of carries either combination. Both intentionally combine units: the terminals feed a generator that composes arbitrary arithmetic, and the composites are numerical features rather than physical quantities. The time-pressure pair normalises the state: HFRAC $= \tau / T_{\max}$, following RemT, and VISFRAC, the fraction of customers already served, after the satisfied-fraction terminal of Liu et al. (2017). Every terminal has a closed-form expression. Where a probability appears, it is a Gaussian tail, never a sample, so a rule composed from any subset can make decisions without simulation.

The provenance question raised in the introduction therefore has a mechanical answer. For any terminal in any rule considered in this paper, the chain reads: the reference routes showed which operations pay, each operation named a concept, and the concept either restores a field term, derives from a published terminal by a stated modification, or corresponds to one of the two composites above. The routes informed what to look for, but no terminal is computed from route data, and no route enters any formula. Whether a terminal earns its place is decided downstream, by the genetic

algorithm of the next subsection, and all eighteen terminals are selected from at least one benchmark instance (Section 4.4).

Table 2 The rule vocabulary: eleven baseline terminals inherited unchanged and eighteen constructed terminals in four families. All are closed form.

Family	Terminal	Definition	Origin	Operational motivation
Inherited	SCORE ... RemT	the ten features of the baseline GPHH	Mei and Zhang (2018), Table I	the raw prize, distance and clock state
	NS	neighbourhood score of the candidate	Jackson and Mei (2020)	prize rate of everything still open after the candidate
Reachability	REACH1/2/3/5	k-step greedy walk, Eq. (3)	NPT-style look-ahead (Zhang et al. 2024); look-ahead pair of Mei and Zhang (2018)	prize reachable within k greedy steps beyond the candidate
	NSTW	NS weighted by the unspent window fraction	variant of NS	reachable value net of the waiting each successor imposes
	NSROB	NS under the σ-stress	pessimistic NS	reachable value that survives a late finish
	REACH2ROB	two-step walk under the σ-stress	pessimistic REACH2	prize reachable in two steps that survives a late finish
	MAXNS	largest term of the NS sum	max-aggregated NS	prize rate of the single best successor
	NFEAS	$\lvert F(v,\tau)\rvert$, Eq. (1)	NIQ analogue (Zhang et al. 2024)	how many customers remain feasible
Window cascade	REGRET, SLOST	prize mass forfeited, Eq. (4); identical by construction	expected regret (Bian and Liu 2018); structural family	prize the commitment forfeits downstream
	NLOST	count of the lost cascade, Eq. (4)	structural regret, count	how many customers the commitment loses
	ATRISK	expected prize at risk, Eq. (5)	chance-constrained SLOST from IAP (Bian and Liu 2018)	prize at risk under the candidate's duration uncertainty
	IAPD	miss probability of the best successor, Eq. (6)	complement of IAP (Bian and Liu 2018)	probability of missing the best downstream successor's window
Composite	LOSSR	SLOST/(NS+1)	novel	cascade damage per unit of reachable value
	NETLOSS	SLOST − NS	novel	downstream loss net of reachable gain
Time pressure	HFRAC	$\tau/T_{\max}$	from RemT (Mei and Zhang 2018)	how much of the horizon is already gone
	VISFRAC	fraction of customers served	satisfied fraction (Liu et al. 2017)	how much of the instance is already served

3.4 Chromosome and fitness

With eighteen candidate terminals, there are 2^{18} possible vocabularies per instance, and Section 2.3 established that evaluating one requires running the full GPHH. In the campaign logs, that unit cost ranges from under a minute on the smallest instances to approximately forty minutes per evaluation on the largest vocabularies of pr14 (Section 4.5). A search requiring a few hundred such evaluations per instance is affordable. This section builds it: a genetic algorithm over terminal subsets whose every fitness evaluation consists of one complete GPHH execution.

The chromosome is a binary vector of eighteen genes, one per constructed terminal, and the eleven baseline terminals stay active in every individual. Two genes, REGRET and SLOST, encode the same function (Section 3.3), so the vocabulary spans seventeen distinct behaviours. Decoding a chromosome hands the generator the union of both sets. Fitness is measured through a full run: sixteen independent GP populations of 1024, one per seed, evolve for 51 generations using the activated vocabulary, each seed returns its best rule, and the sixteen rules are re-scored on a fixed train block of 64 scenarios drawn once with seed 2 000 000. The fitness of the chromosome is the mean of those sixteen train scores. The inner GP configuration remains unchanged, *i.e.*, the generator keeps the exact baseline configuration of Section 4.1, with only its vocabulary varying across chromosomes.

Three scenario layers keep the search honest. The inner GP trains on its own rotating samples, the wrapper compares chromosomes on the fixed train block, and the held-out set of 500 scenarios is not touched until every choice is closed. Sixteen seeds per evaluation, approximately half the final protocol, is a cost decision, and Section 3.6 describes the correction for it. Evaluations run in lockstep on the GPU, with batches of up to thirteen chromosomes per launch under per-genome terminal masks, and a fixed search seed with a persistent cache of per-seed fitness vectors ensures deterministic and resumable execution.

3.5 The structured population

The population is organised as a ternary tree of thirteen individuals, an arrangement with a long lineage in memetic algorithms (Moscato 1989; Moscato and Cotta 2003; Toledo et al. 2014) and at exactly this size, documented as a good compromise between solution quality and computational effort (Mendes et al. 2005; de Assis et al. 2015, 2019). The thirteen individuals form four overlapping clusters, each consisting of one leader with three supporters: the root leads the three mid-level nodes, and each mid-level node leads three leaves. Whenever a supporter beats its leader, the two swap places, so the best chromosome is always located at the root (Figure 3). Structured populations control which individuals recombine (Lim 2014), and here this structure defines the whole reproduction scheme.

Initialisation is informed rather than random. The empty chromosome and the eighteen singletons are evaluated, and the best thirteen of those nineteen become the tree. The full chromosome is excluded from seeding, so every run starts from small vocabularies and expands them. A pilot with random initialisation underperformed

the singleton-seeded runs on three of four comparable instances and was therefore abandoned. Both sets of runs are available in the repository.

Reproduction is deterministic and follows the clusters (Toledo et al. 2014): each leader crosses with each of its three supporters, producing twelve children per generation. The crossover is uniform (Syswerda 1989), which fits the object under search. Genes shared by both parents are inherited unchanged, while genes in the symmetric difference are selected with equal probability, thus an offspring retains the common core of its parents (Figure 4). The figure draws leader [MAXNS, NSTW, REACH3, REACH5] against supporter [REACH5, REACH2ROB], and the child, after one flip, is [REACH5, REACH2ROB], the confirmed champion of pr14. The draw is illustrative, the chromosomes are real. Each offspring then undergoes bit-flip mutation with probability 0.5/18 per gene (Bäck 1993), serving as background variation rather than a primary search mechanism. An offspring replaces the worse of its two parents when it improves upon it. When the population collapses to six or fewer distinct chromosomes, two random chromosomes are reinjected into the bottom leaves, following the small-population reset used in micro-genetic algorithms (Krishnakumar 1989). The search stops after ten generations without improvement of the best individual, with a fifty-generation cap as a safeguard that no campaign run ever reached. Algorithm 1 summarises these stages.

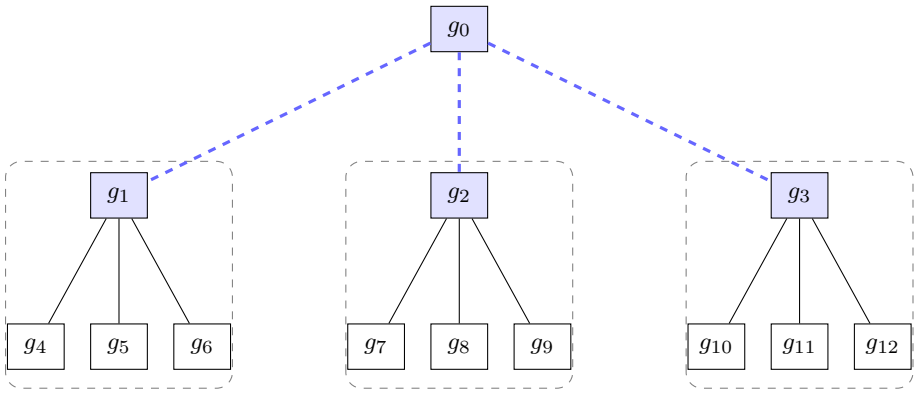


Fig. 3 The structured population: thirteen chromosomes on a ternary tree of four overlapping leader–supporter clusters. Three clusters are the dashed boxes; the fourth is the root with the three mid-level nodes (thick dashed edges).

3.6 Confirmation and the cross-validatory choice

Sixteen seeds provide an effective search signal, but the pr14 log shows why they are insufficient for the final call. The finalists of a run, the thirteen tree slots deduplicated and sorted with ties broken toward fewer genes, have their top six re-evaluated using all thirty seeds. On pr14 this confirmation reordered the ranking: the proxy leader decreased from 1220.9 to 1214.4, while [REACH5, REACH2ROB], fourth by proxy at 1217.8, confirmed at 1218.4 and became the champion. Every instance in the campaign retained its confirmed champion.

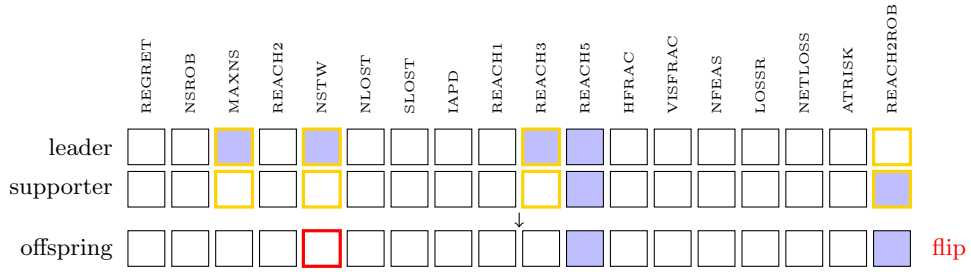


Fig. 4 One reproduction, drawn with two pr14 finalists. Blue cells are active genes, yellow borders the divergent genes drawn at fifty-fifty, red the mutation flip.

Algorithm 1 Per-instance terminal-subset search

evaluate the empty chromosome and the 18 singletons ▷ each evaluation: one full GPHH run, cached
the best 13 form the ternary tree
for generation 1 to 50 **do**
    if distinct chromosomes ≤ 6 **then**
        reinject 2 random chromosomes into the bottom leaves
    end if
    for each cluster, each of its 3 supporters **do**
        child $\leftarrow$ mutate(uniform_crossover(leader, supporter))
        evaluate child
        if child improves on the worse of its two parents **then**
            replace that parent with the child
        end if
    end for
    restore the hierarchy: swap any supporter above its leader
    if 10 generations without a new best **then break**
    end if
end for
deduplicate the finalists, confirm the top 6 at 30 seeds
champion $\leftarrow$ best confirmed mean, ties toward fewer genes
cross-validatory choice between champion and NS-GP on the train block, held-out read once

The deployed rule is then decided by cross-validatory choice (Stone 1974). Specifically, the thirty final rules of the champion and the thirty NS-GP baseline rules are evaluated on the same train block, the member with the better mean train score is selected, and the held-out set is read once for reporting. The baseline is always included as a candidate, in the spirit of the `SATzilla` backup (Xu et al. 2008), ensuring that the deployed rule cannot have a lower training score than NS-GP by construction. The resulting held-out performance is the subject of Section 4.

The wrapper cost is documented in the published campaign logs: a median of 113 full GPHH executions per instance, ranging from 65 to 295, and over 5000 in the

whole campaign, all paid once, at design time. The next section specifies the evaluation protocol used for all subsequent results.

4 Results

4.1 Experimental setup

All experiments use the forty benchmark instances of the stochastic TOPTW, five per dataset from the `Solomon` `c`, `r`, and `rc` groups at both scheduling horizons and from the Cordeau `pr` set, following the selection introduced by Mei and Zhang (2018) and retained by subsequent studies. Instance sizes range from approximately 50 to 288 customers, with a fleet of three vehicles. Every deterministic duration $d(p)$ becomes the random variable $D(p) \sim \mathcal{N}(d(p), \sigma\, d(p))$, with mean and standard deviation defined as in the lineage, using the benchmark uncertainty level $\sigma = 0.2$ (Mei and Zhang 2018), truncated at zero in our implementation (Section 3.3).

Every reported score is the mean over 30 independent runs of the generator, and each run is evaluated once on a held-out set of 500 scenarios, the sample size used by the lineage for a precise approximation of expected performance (Mei and Zhang 2018). The 500 scenarios are drawn with a fixed seed and shared across all methods and every rule, providing common random numbers and enabling paired comparisons by seed index in Section 4.3. The thirty runs measure the training procedure itself. Deploying a single rule would mean selecting one run by its train score, never by the held-out read. No evolved rule accesses the held-out set before the final campaign, which reads it exactly once per experiment. The same set is used earlier, in Section 3.1, to replay the frozen reference routes as a design-time measurement of fixed policies, with no terminal, parameter, or choice tuned on the draws. Training, by contrast, follows the instance-rotation protocol of Mei and Zhang (2018): a single fresh sample is drawn per generation using the run's seed.

The NS-GP baseline is re-evolved with our engine under exactly the configuration of Table 3. Re-evolving it, rather than transcribing published scores, places both sides of every comparison on the same engine, scenarios, and hardware. The engine is verified bit-exact between its GPU and CPU paths by the six gates shipped with the repository. The campaign ran on three consumer NVIDIA GPUs, an RTX 5090, an RTX 4090 and an RTX 5060 Ti. Every timing figure comes from the RTX 5090 alone. Runtime results are reported in Section 4.5. Table 3 summarises the setup: the generator block is the baseline configuration of the lineage unchanged, while the wrapper block consolidates Sections 3.4 to 3.6 in one place.

4.2 Experimental Results

Table 4 reports the held-out outcome on the forty instances, for both methods presented as mean and standard deviation over the same thirty paired runs. SI-GP improves upon the NS-GP baseline by 1.0% on average, with 36 wins, 4 ties, and no losses. The deployed rule is the evolved rule on thirty-eight instances. On pr13 and rc205, cross-validatory selection instead retained the baseline, and these instances

Table 3 Every parameter of the pipeline. The generator block is identical to the NS-GP baseline configuration; the wrapper block summarises Sections 3.4 to 3.6.

	Parameter	Value	Source
Problem	benchmark instances	40, five per dataset	(Mei and Zhang 2018; Jackson and Mei 2020)
	vehicles	$m = 3$	(Mei and Zhang 2018)
	uncertainty	$D(p) \sim \mathcal{N}(d(p), \sigma d(p))$, $\sigma = 0.2$	(Mei and Zhang 2018)
	held-out evaluation	500 scenarios, common random numbers, read once	(Mei and Zhang 2018)
	runs per configuration	30 independent	(Mei and Zhang 2018)
Generator (GP)	population, generations	1024, 51	(Mei and Zhang 2018)
	initialisation	ramped half-and-half, maximal depth 8	(Mei and Zhang 2018)
	function set	$+$, $-$, $\times$, protected $\div$ ($= 1$ at zero), min, max	(Mei and Zhang 2018)
	selection, elitism	tournament 7, elitism 10	(Mei and Zhang 2018)
	crossover, mutation, reproduction	0.80, 0.15, 0.05	(Mei and Zhang 2018)
	training samples	1 per generation, instance rotation	(Mei and Zhang 2018)
Wrapper (GA)	population	13, ternary tree	(Toledo et al. 2014)
	crossover, mutation	uniform, 0.5/18 per gene	(Syswerda 1989; Bäck 1993)
	fitness seeds, confirmation seeds	16, 30	Sect. 3.4
	fitness train block	64 scenarios, fixed seed	Sect. 3.4
	stop, guard	10 stagnant generations, 50 cap	Sect. 3.4
	finalists, reinjection	top 6 confirmed, 2 random at ≤ 6 distinct	Sect. 3.4

contribute zero to all aggregates, as implied by the backup construction. Ten improvements remain significant after Holm correction, marked by bullets in the table, and all ten have large effect sizes (Section 4.3).

By analysing the results obtained, it is possible to observe that the gains follow the geography of the reference gap. The random short-horizon family averages +2.8%, including the two largest gains, r105 at +3.6% and r103 at +3.5%. The mixed and clustered short-horizon families average +1.5% and +1.3%, respectively, while the Cordeau groups average +1.2% and +0.8%. The long-horizon families show smaller gains, between +0.1% and +0.3%, where the baseline already serves nearly every customer and Section 4.4 indicates there is nothing left to collect. Excluding the ten instances the baseline saturates outright, the average gain is 1.3%. Figure 5 orders the forty official gains, while Figure 6 shows the per-seed distributions by family.

Two internal anchors further contextualise the results. The lower one is NS-GP itself: the eleven baseline terminals stay active in every individual, so the empty chromosome is the baseline, available on every instance and scoring zero because every gain is measured against it. That is what carries the never-trains-below construction

of Section 3.6 to the held-out side as never worse. The upper one is the virtual best of the per-instance portfolio, the member that would be selected with held-out access, which also averages +1.0%. The train-side choice matches it on thirty-nine of forty instances, and on the exception, r204 (Section 4.3), the two differ by 0.03 points.

Figure 7 shows the search trajectory through the family-mean best-so-far train gain per GA generation, starting from individuals with a single active gene. Families that achieve held-out gains are those that exhibit sustained improvement, whereas the long-horizon families show limited progression, and every run stopped due to stagnation. The wrapper searched and converged, and the gains appeared where the reference gap of Section 3.1 indicated there was something to find.

An ablation isolates the two halves of the method on the ten instances of Table 5. The set comprises the strongest SI-GP cases, therefore, the analysis is conditional on that choice, rather than an estimate of the average selection benefit over the benchmark, and its p-values are exploratory. Handing the generator the whole eighteen-terminal vocabulary, without selection, lifts the baseline by 1.31% on average and never underperforms it, a significant paired win on seven of the ten instances. The elected subsets average 2.36% on the same ten instances. They sit above the unrestricted vocabulary on nine, five of the paired differences are significant, and the sole exception, c101 at −0.23%, is not statistically distinguishable from zero ($p = 0.88$). pr01 provides the clearest contrast: the unrestricted vocabulary ties the baseline there ($p = 0.80$) and the elected subset carries the whole +2.85%. On these ten, vocabulary selection concentrates the search in ways that the unrestricted vocabulary does not. The ten ablation runs cost 36 minutes on the RTX 5090, and their traces and paired statistics are included in the repository.

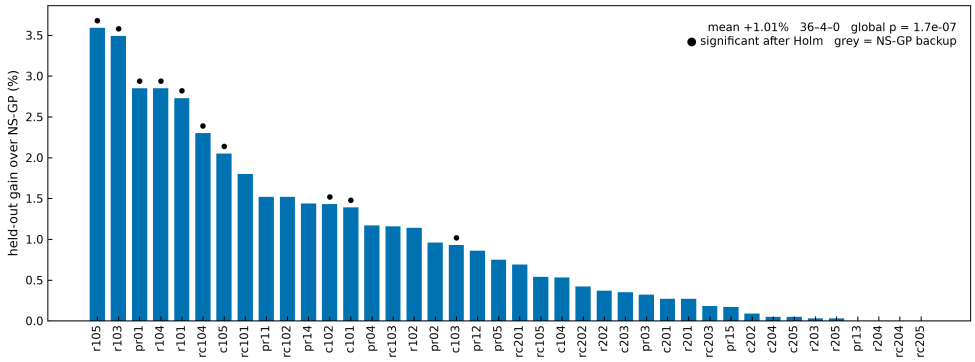


Fig. 5 The forty official held-out gains over NS-GP, ordered. Dots mark the ten instances significant after Holm correction. The four zeros on the right are the two backup instances and the two ties.

4.3 Statistical analysis

Every test in this section uses the same thirty paired runs of Section 4.1. Because the two methods share seeds and scenarios, the natural per-instance test is the Wilcoxon

Table 4 Held-out results on the forty instances: mean (std) over the thirty paired runs of each side, the official gain (backup instances enter as zero), and the deployed rule. • marks gains that survive Holm correction (Section 4.3).

Instance	NS-GP	SI-GP	Gain (%)	Deployed
c101	763.92 (14.40)	774.57 (14.78)	+1.39•	SI-GP
c102	860.52 (11.06)	872.86 (11.36)	+1.43•	SI-GP
c103	906.09 (10.70)	914.49 (4.13)	+0.93•	SI-GP
c104	953.03 (10.24)	958.06 (7.83)	+0.53	SI-GP
c105	807.85 (18.71)	824.39 (6.83)	+2.05•	SI-GP
c201	1779.29 (12.22)	1784.14 (3.67)	+0.27	SI-GP
c202	1795.22 (7.01)	1796.87 (5.03)	+0.09	SI-GP
c203	1785.05 (19.06)	1791.28 (8.20)	+0.35	SI-GP
c204	1798.44 (6.10)	1799.34 (7.35)	+0.05	SI-GP
c205	1802.82 (5.64)	1803.68 (4.42)	+0.05	SI-GP
r101	441.14 (7.43)	453.18 (6.75)	+2.73•	SI-GP
r102	616.02 (13.92)	623.07 (9.37)	+1.14	SI-GP
r103	650.62 (20.27)	673.33 (10.91)	+3.49•	SI-GP
r104	709.54 (15.74)	729.78 (11.12)	+2.85•	SI-GP
r105	548.58 (27.06)	568.30 (9.18)	+3.59•	SI-GP
r201	1369.84 (20.92)	1373.50 (12.54)	+0.27	SI-GP
r202	1427.25 (12.40)	1432.49 (9.77)	+0.37	SI-GP
r203	1456.39 (1.19)	1456.81 (0.96)	+0.03	SI-GP
r204	1457.96 (0.11)	1457.93 (0.10)	+0.00	SI-GP
r205	1456.85 (1.52)	1457.28 (0.72)	+0.03	SI-GP
rc101	561.72 (14.45)	571.85 (9.28)	+1.80	SI-GP
rc102	618.53 (15.33)	627.96 (11.94)	+1.52	SI-GP
rc103	667.89 (14.77)	675.63 (16.82)	+1.16	SI-GP
rc104	728.42 (15.71)	745.14 (13.65)	+2.30•	SI-GP
rc105	616.16 (8.51)	619.46 (11.31)	+0.54	SI-GP
rc201	1592.88 (34.34)	1603.91 (9.47)	+0.69	SI-GP
rc202	1661.76 (14.78)	1668.79 (11.91)	+0.42	SI-GP
rc203	1715.02 (8.03)	1718.07 (4.47)	+0.18	SI-GP
rc204	1723.89 (0.36)	1723.93 (0.25)	+0.00	SI-GP
rc205	1627.95 (31.13)	1627.95 (31.13)	+0.00	NS-GP
pr01	570.51 (10.84)	586.78 (6.63)	+2.85•	SI-GP
pr02	853.61 (13.81)	861.81 (12.87)	+0.96	SI-GP
pr03	899.89 (17.86)	902.75 (20.17)	+0.32	SI-GP
pr04	1136.79 (27.18)	1150.08 (18.51)	+1.17	SI-GP
pr05	1301.60 (21.27)	1311.41 (28.63)	+0.75	SI-GP
pr11	622.07 (13.28)	631.53 (9.22)	+1.52	SI-GP
pr12	910.45 (18.14)	918.27 (17.48)	+0.86	SI-GP
pr13	1046.43 (14.24)	1046.43 (14.24)	+0.00	NS-GP
pr14	1201.22 (21.29)	1218.53 (26.06)	+1.44	SI-GP
pr15	1477.67 (22.30)	1480.22 (25.56)	+0.17	SI-GP
mean			+1.01	38 SI-GP, 2 NS-GP

signed-rank test on the thirty per-seed differences, a paired test that exploits common random numbers and is more powerful in this setting than the unpaired rank-sum used in the lineage (Mei and Zhang 2018; Jackson and Mei 2020). The forty per-instance tests constitute a single family; their p-values are therefore adjusted using the sequentially rejective correction of Holm (1979). Alongside every p-value we report the Vargha and Delaney (2000) A_{12} effect size, representing the probability that a random SI-GP run outperforms a random baseline run on the same instance.

At $\alpha = 0.05$, the paired test identifies significant improvements on 18 of the 38 evolved instances for which the evolved rule is deployed. One additional instance, r204, shows a significant difference in the opposite direction, with 3 wins and 27 losses at

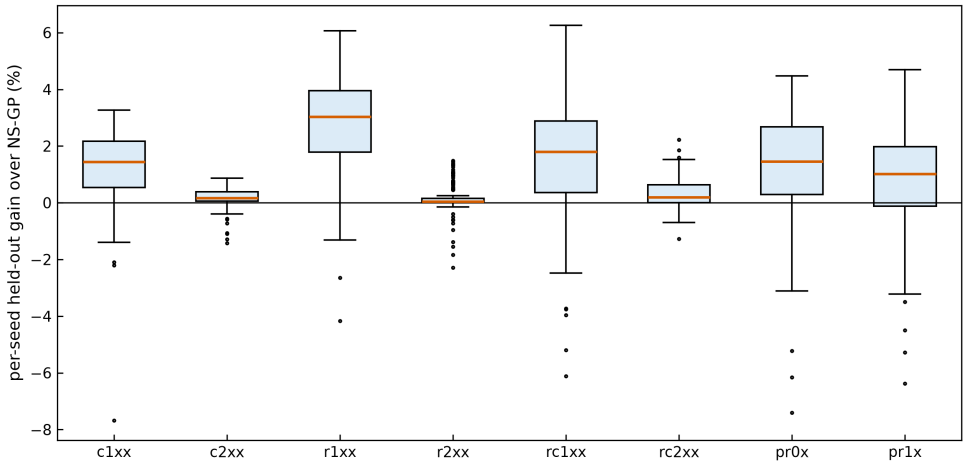


Fig. 6 Per-seed held-out gains of the thirty evolved rules over the NS-GP mean, by instance family. The short-horizon and Cordeau families carry the gains, the long-horizon families sit at zero.

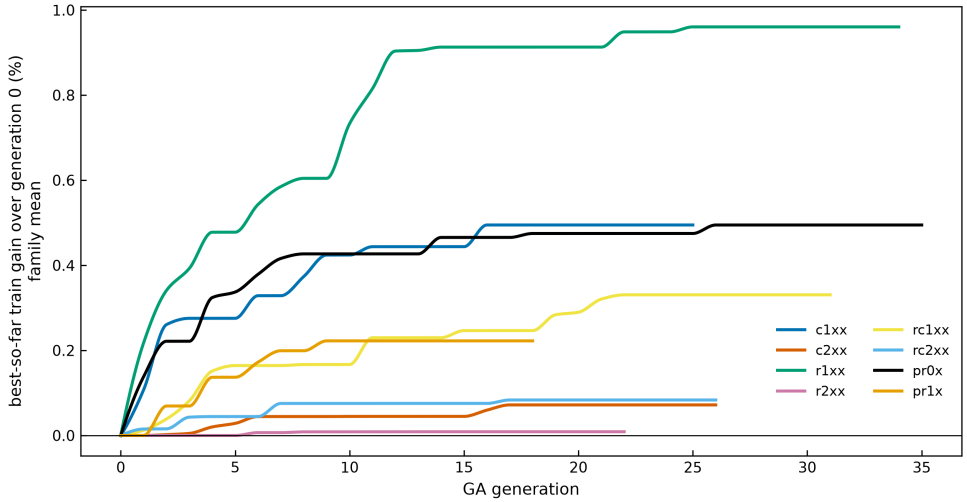


Fig. 7 Family-mean best-so-far train gain per GA generation, from the singleton-seeded start. Curves end where the last run of the family stopped by stagnation.

$p = 1.6 \times 10^{-4}$, despite a difference of only 0.03 points relative to a mean near 1458. Table 4 rounds official gains to two decimals and therefore reports this case as a draw. Treating it as a loss would move the 36–4–0 count but not any mean, test statistic, or effect size, all of which are computed from the same thirty paired differences. After Holm correction, ten improvements remain significant, all with large effects, $A_{12} \geq 0.71$

(Table 5). The complete forty-line instance table, including all p-values, corrections, and effect sizes, is provided in the repository.

Following Demšar (2006), we also perform a global Wilcoxon signed-rank test over the forty per-instance mean differences. It rejects the null hypothesis at $p = 1.7 \times 10^{-7}$, with 36 wins, 4 draws, and no losses. For comparability, we additionally apply the lineage's unpaired rank-sum criterion at $\alpha = 0.05$; it marks 19 of 38 evolved instances, one more than the paired count, including rc204, whose official gain is a literal zero. With such small performance differences, statistical significance does not necessarily imply practical relevance, motivating the inclusion of an effect size with every statistical claim in this paper.

The remaining guarantees are structural. The deployed selection never achieves a lower training score than the baseline (Section 3.6) and was also never worse in the held-out evaluation. The two backup instances contribute zero to every aggregate, as specified in Table 4. Every score is a mean over the thirty runs, never a best of any number of them, and the evolved rules access the held-out scenarios exactly once, after all design decisions are finalised. The earlier evaluations in Section 3.1 concern fixed policies that were not calibrated using held-out draws and provide no parameters to the proposed method.

Table 5 The ten instances whose paired gains survive Holm correction over the family of forty tests. A_{12} is the probability that a random SI-GP run beats a random NS-GP run of the same instance; all ten are large effects.

Instance	**Gain (%)**	p	p_{Holm}	A_{12}
c105	+2.05	5.6×10^{-9}	2.2×10^{-7}	0.91
pr01	+2.85	1.6×10^{-7}	6.2×10^{-6}	0.90
r103	+3.49	4.4×10^{-6}	1.6×10^{-4}	0.87
r101	+2.73	5.1×10^{-6}	1.9×10^{-4}	0.89
r104	+2.85	5.1×10^{-6}	1.9×10^{-4}	0.93
r105	+3.59	9.2×10^{-6}	3.1×10^{-4}	0.84
c102	+1.43	3.9×10^{-5}	1.3×10^{-3}	0.81
rc104	+2.30	2.1×10^{-4}	6.5×10^{-3}	0.83
c101	+1.39	5.1×10^{-4}	1.5×10^{-2}	0.81
c103	+0.93	6.1×10^{-4}	1.8×10^{-2}	0.75

4.4 Interpretability and mechanism

Figure 8 shows the terminal subsets selected by the search for each instance, with both axes grouped by family. The subsets are small, ranging from one to seven terminals with a median of two, and thirteen instances elect a single constructed terminal. Reachability dominates: REACH2 appears in 13 of the 39 winning subsets, VISFRAC in 11, REACH3 and IAPD in 10 each, and all eighteen terminals are selected in at least one subset. The identity of Section 3.3 is also reflected in the results, *i.e.* REGRET appears in one subset (r101), SLOST in two (c104, rc102), and no subset elects both. The ten Cordeau instances, introduced after the vocabulary was fixed, elect the same

terminal families identified by the `Solomon` analysis, with look-ahead predominating, and risk and cascade following. Two columns need a note: the pr13 subset was subsequently discarded by cross-validation, while rc205 retained the baseline, yielding an empty column.

Figure 9 examines one champion, the pr14 rule at its best held-out seed, selected retrospectively for interpretation, +4.7% over the baseline mean in thirty-seven nodes. Its selected subset is [REACH5, REACH2ROB], and the tree illustrates what election does and does not decide: this seed composes REACH5 three times, twice amplifying the candidate's prize in the numerator and once contributing to the time normalisation, while REACH2ROB is never invoked. Election bounds what the GP may use, while composition stays free. The resulting rule can be interpreted as a prize-per-travel score scaled by the five-step reachable prize, corresponding to the radial reach shown for the pr14 in Figure 12.

Table 4 provides evidence for the mechanism behind the aggregate: on 29 of the 38 evolved instances, SI-GP has a lower standard deviation than the baseline, with rc201 showing an almost fourfold reduction. Figure 10 further characterises this variance. For each instance, we correlate the per-seed gain with the held-out score of the paired baseline run. All thirty-nine correlations are negative, with median -0.80, indicating that the evolved rules gain most where the baseline run converged worst. Part of that slope is arithmetic, since gains computed relative to the corresponding baseline lean negative whenever the baseline varies, so we read these correlations as supporting evidence alongside the variance results rather than as independent proof. With the constructed terminals included in the primitive set, fewer runs converge to fragile rules, which fits a vocabulary regularising the GP search rather than merely increasing prize. r105 shows both halves at once. Its weakest baseline seeds are rescued outright, the per-seed floor rises from 438.7 to 548.4, and the twenty-seven seeds outside the rescue still gain +2.3% on average, 22 wins to 5 losses, quality under noise rather than the repair of a few bad runs. c203, deep in the saturated family, shows the regularisation effect more directly: 91% of its small gain comes from the three weakest baseline seeds, the standard deviation across seeds falls from 19.1 to 8.2, and the other twenty-seven seeds break even. This pattern is not universal. On pr05, pr14, and pr15, the standard deviation rises with the mean, indicating a trade-off between variance and prize, so there is a tendency for the regularising signature to be strongest on the `Solomon` groups.

The magnitude of the gain is associated with a single static quantity. Figure 11 plots the official gain against the coverage of the best baseline seed decoded at $\sigma = 0$, the fraction of customers NS-GP already serves without noise. The ten grey instances at 100.0% correspond to the long-horizon groups in Table 4: the baseline already serves every customer, leaving no additional prize to collect, so gains are limited by stabilisation effects. On r204 and rc204, the thirty runs of both sides land within a fraction of a point, yielding a literal zero gain, while c201 and c203 retain stochastic variation across seeds and obtain small positive gains consistent with partial variance reduction. Below saturation, every `Solomon` instance with a gain above 2% has coverage between 20% and 36%, but coverage alone is not predictive: pr13, at 45.1%, is the only instance whose evolved rule is discarded, while pr01 gains +2.9% at 85.4% and pr11 +1.5% at

93.8%. Low coverage therefore indicates that a gain is available, not that a subset will find it.

Figure 12 presents the geometric interpretation, with one instance per benchmark family and the deployed rule shown alongside the paired baseline and reference route. In c102, the reference partitions the plane into sectors, and the evolved rule recovers the partition where the baseline tangles its three tours through the centre. The r103 baseline abandons the eastern side of the map entirely, exactly the sector the reference serves, and the evolved tours extend eastward. In rc104, the reference assigns one vehicle to the distant southern cluster, a structure also recovered by SI-GP. pr14 is the starkest: the baseline remains concentrated in the dense centre whereas both the reference and the evolved rule extend outward. In the saturated c205, the control, the three columns nearly coincide. No rule ever saw a route. The route patterns re-emerge because the terminals that survived election are the ones that name them.

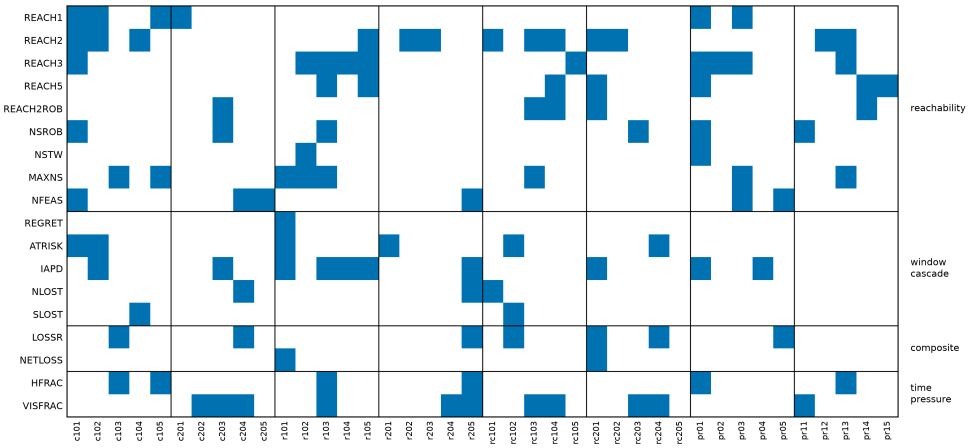


Fig. 8 The winning terminal subsets of the per-instance search, terminals by instances, both grouped by family.

4.5 Computational cost

Table 6 quantifies the fundamental computational unit underlying the experiments: one complete GPHH execution, thirty seeds of a 1024-individual, 51-generation run with instance rotation, timed on the RTX 5090 (Section 4.1). Under the baseline vocabulary the unit costs 21 seconds on c101 and 188 on pr15. Four constructed terminals raise this to 36 and 1092 seconds, corresponding to factors of 1.7 and 5.8 that grow with instance size. This increase arises because the reachability and cascade families enumerate open successors at every decision, the same enumeration that made the look-ahead variant of the lineage "much less efficient" (Mei and Zhang 2018). The held-out column quantifies the evaluation cost. Scoring all thirty rules from a run across 500 scenarios requires between 0.1 and 12.5 seconds.

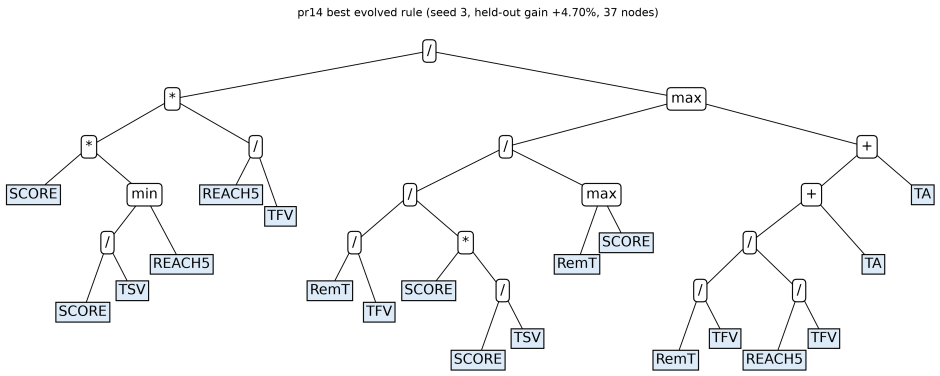


Fig. 9 The pr14 rule at its best held-out seed, thirty-seven nodes.

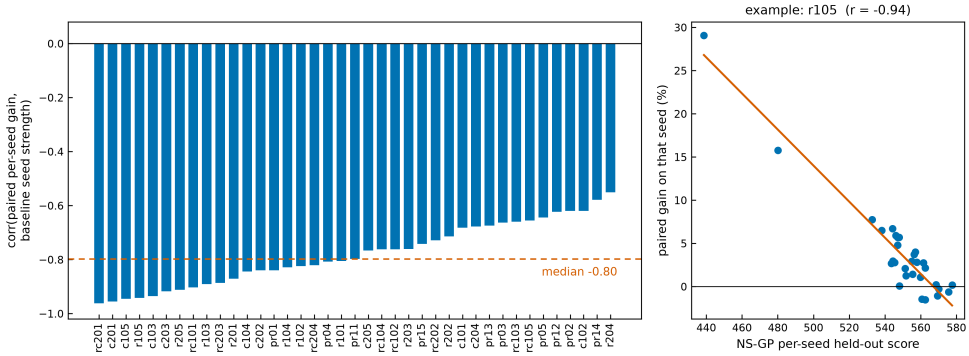


Fig. 10 Left: per-instance correlation between the per-seed gain and the held-out score of the paired baseline run. Right: the strongest case, r105.

Hardware and implementation separate the two engines by orders of magnitude, `ECJ` executes on a CPU, whereas the proposed implementation uses CUDA on a consumer GPU. The comparison of Section 2.6 against Table 6 is structural rather than numerical: vocabulary classes that were previously computationally prohibitive now require minutes, reversing the asymmetry that discouraged rich terminals. This reduction makes the wrapper affordable at all. Each chromosome evaluation requires one full GPHH execution at sixteen seeds, and each confirmation uses thirty seeds. The campaign incurred a median of 113 executions per instance (Section 3.6). The computational burden is most pronounced in the published logs, *i.e.* each confirmation evaluation on pr14 required 2517 seconds, and confirmation of its six finalists took slightly over four hours, wall clock time. The reference computation was likewise performed once: the `cuOpt` routes behind the gap of Section 3.1 used a 900-second budget per instance, with `PyVRP` supplying the winner on five instances.

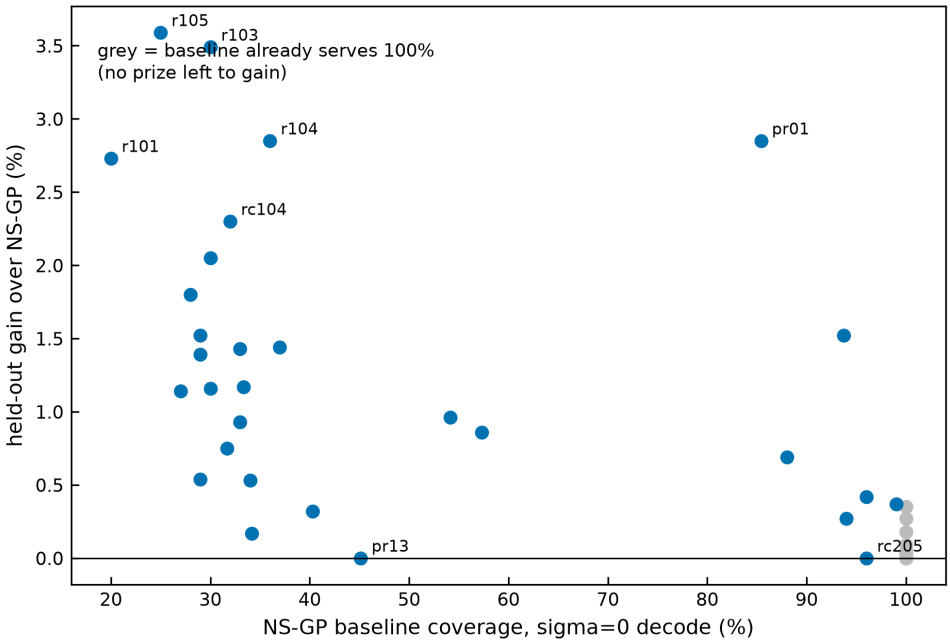


Fig. 11 Official held-out gain against the coverage of the best NS-GP seed decoded at $\sigma = 0$. Grey marks the ten instances the baseline already serves completely.

These costs are incurred entirely at design time. The deployed artefact is a tree such as that in Figure 9, deciding the next customer evaluates one arithmetic expression per open candidate, without a scenario sampled or a solver called. Thus the whole design cost, GP, GA, confirmation, and reference routes together, is incurred before the first deployed decision.

Table 6 Wall-clock seconds on the RTX 5090 for one complete GPHH execution under the eleven-terminal baseline vocabulary and with four constructed terminals added, and for the held-out evaluation of all thirty resulting rules.

		Train		Held-out	
Instance	**Customers**	**11 terms**	**15 terms**	**11 terms**	**15 terms**
c101	100	21	36	0.1	0.3
r201	100	69	134	0.3	1.0
pr15	240	188	1092	1.7	12.5



Fig. 12 One instance per benchmark family, rows c102, r103, rc104, pr14 and c205. Columns: the NS-GP tour, the winning reference route, and the SI-GP tour, the two rules decoded at $\sigma = 0$ at their best held-out seeds, chosen for the picture only, with three vehicles. Small dots are unserved customers.

5 Conclusions

This paper proposed a method for evolving interpretable dispatching rules that capture part of the gap to high-quality offline solvers while making deployment decisions

without any simulation. On the forty-instance stochastic TOPTW benchmark, the deployed rules improve the re-evolved NS-GP baseline by 1.0% on average and by 1.3% outside saturation, without underperforming on any instance. Ten instances remain significant after Holm correction, all with large effects. From the results, it is possible to conclude that the average deployed gain recovers approximately one quarter of the measured reference gap. The three objectives of the study were addressed sequentially. Route analysis produced 18 constructed terminals, 16 grounded in published terminals, and two novel composites, with an explicit chain from route operation to literature source and election record. The structured-population GA made per-instance selection affordable, requiring a median of 113 full GPHH executions per instance, and selecting compact vocabularies, with a median size of two terminals. The paired evaluation protocol validated the gains, while Section 4.4 provided an explanation: the evidence is consistent with the vocabulary regularising the GP search, cutting the run-to-run deviation and improving weak seeds where the baseline converges worst, and the $\sigma = 0$ coverage of an instance indicates the remaining potential for prize collection.

The clearest limitations of SI-GP are the three instances for which the pipeline did not deploy an improvement. On pr13, the subset selected by the search underperformed the baseline on the training choice; consequently, the backup rule was deployed, and the held-out evaluation confirmed the choice, with the evolved rule obtaining -0.1%. On rc205, the search itself returned the bare baseline. Both instances contribute zero to the aggregates rather than losses, as intended by the backup mechanism, although their reference gaps are $+7.9\%$ and $+3.5\%$, respectively, indicating prize that high-quality solvers can collect but current vocabulary cannot encode. pr15 provides an even stronger example. Despite exhibiting the second largest gap in the benchmark ($+11.1\%$), its champion is the REACH5 singleton, and the deployed gain is $+0.2\%$. The wrapper converged there, with every run stopped by stagnation. This suggests that the limitation lies not in search effort, but in the expressive power of the current vocabulary or rule representation.

The evidence presented is restricted to a single problem domain: forty instances of the stochastic TOPTW, the three-vehicle fleet, the uncertainty level $\sigma = 0.2$ inherited from the lineage, and one held-out evaluation protocol. The proposed pipeline is not specific to orienteering problems: reference routes, route reading, literature grounding, and per-instance selection define a procedure that can transfer to other domains. Nevertheless, all quantitative results reported here are specific to TOPTW. The benchmark also illustrates where computational effort is justified. Ten of the forty instances are saturated by the baseline, and their searches returned at most $+0.35\%$. The millisecond $\sigma = 0$ decode of Figure 11 identifies such cases in advance. They were included here to support the never-worse claim, but practical applications should prioritise instances with remaining improvement potential. The vocabulary factor described in Section 4.5 also increases with instance size, making exhaustive per-instance searches on substantially larger problems computationally demanding. Two wrapper controls remain unexplored: a random search over subsets under the same evaluation budget and a flat genetic algorithm of equal size. The random-initialisation pilot of Section 3.5 and the published stagnation ablation provide partial evidence on this question, but the specific contribution of the structured population is not isolated, and is not measured here.

Three directions emerge for future research. The first is a static per-instance selector. The campaign produced forty elected subsets and associated instance descriptors, namely the $\sigma = 0$ coverage, enabling the development of predictive models that map instance features to terminal subsets. Such models could provide immediate recommendations, support informed initialisation of the GA, or act as portfolio selectors over evolved champions. The second direction is the remaining reference gap, particularly on instances such as pr15, where additional route analysis may reveal operations not captured by the current vocabulary. The third is transfer to neighbouring GPHH domains, including dynamic job shop scheduling and uncertain arc routing, where high-quality offline solvers are already available.

The engine, benchmark instances, baseline runs, reference routes, traces, and complete per-instance statistical table are publicly available. The implementation is verified bit-exact between its CPU and GPU paths, and every number in this paper can be reproduced from the released artefacts. However, the deployed solution remains compact. An arithmetic tree with a few dozen nodes per instance, evaluated for each candidate in microseconds, without solver calls or scenario sampling. The complete decision policy can therefore be embedded directly into operational systems, which is precisely the motivation that originally inspired this work.

Acknowledgements.

Declarations

Funding. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001; by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), under Grants No. 312165/2025-0 and No. 102199/2024-8; and by the Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ), through the Jovem Cientista do Nosso Estado Program (JCNE 2025), under Process No. SEI-260003/020882/2025.

Competing interests. The authors declare no competing interests.

Author contributions. A. Mendonça and F. Sousa: conception and experiments. I. Coelho and L. Assis: supervision and review.

Use of AI-assisted tools. We disclose that Writefull for Overleaf was used for orthographic revision. All scientific content and methodology remain the authors' responsibility.

Data availability. The benchmark instances and all campaign results are available in the public repository referenced below.

Code availability. The C++20/CUDA engine, all campaign artefacts and the verification gates are at https://github.com/augustompm/solver-informed-dispatching-rules.

References

Acero F, Li Z (2024) Distilling reinforcement learning policies for interpretable robot locomotion: Gradient boosting machines and symbolic regression. arXiv preprint https://doi.org/10.1109/IROS58592.2024.10802433, arXiv:2403.14328

de Assis LS, González JFV, Usberti FL, et al (2015) Switch allocation problems in power distribution systems. IEEE Transactions on Power Systems 30(1):246–253. https://doi.org/10.1109/TPWRS.2014.2322811

de Assis LS, de P. Junior JR, Tarrataca L, et al (2019) Efficient Volterra systems identification using hierarchical genetic algorithms. Applied Soft Computing 85:105745. https://doi.org/10.1016/j.asoc.2019.105745

Bäck T (1993) Optimal mutation rates in genetic search. In: Proceedings of the 5th International Conference on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp 2–8

Baeta F, Correia J, Martins T, et al (2021) TensorGP – genetic programming engine in TensorFlow. In: Applications of Evolutionary Computation (EvoApplications 2021), Lecture Notes in Computer Science, vol 12694. Springer, Cham, pp 763–778, https://doi.org/10.1007/978-3-030-72699-7_48

Bian Z, Liu X (2018) A real-time adjustment strategy for the operational level stochastic orienteering problem: A simulation-aided optimization approach. Transportation Research Part E: Logistics and Transportation Review 115:246–266. https://doi.org/10.1016/j.tre.2018.05.004

Bischl B, Kerschke P, Kotthoff L, et al (2016) ASlib: A benchmark library for algorithm selection. Artificial Intelligence 237:41–58. https://doi.org/10.1016/j.artint.2016.04.003

Burke EK, Hyde MR, Kendall G (2012) Grammatical evolution of local search heuristics. IEEE Transactions on Evolutionary Computation 16(3):406–417. https://doi.org/10.1109/TEVC.2011.2160401

Burke EK, Gendreau M, Hyde M, et al (2013) Hyper-heuristics: a survey of the state of the art. Journal of the Operational Research Society 64(12):1695–1724. https://doi.org/10.1057/jors.2013.71

Campbell AM, Gendreau M, Thomas BW (2011) The orienteering problem with stochastic travel and service times. Annals of Operations Research 186:61–81. https://doi.org/10.1007/s10479-011-0895-2

Chao IM, Golden BL, Wasil EA (1996) The team orienteering problem. European Journal of Operational Research 88:464–474. https://doi.org/10.1016/0377-2217(94)00289-4

Cordeau JF, Gendreau M, Laporte G (1997) A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 30(2):105–119. https://doi.org/10.1002/(sici)1097-0037(199709)30:2⟨105::aid-net5⟩3.0.co;2-g

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7:1–30

Drake JH, Kheiri A, Özcan E, et al (2020) Recent advances in selection hyper-heuristics. European Journal of Operational Research 285(2):405–428. https://doi.org/10.1016/j.ejor.2019.07.073

Federici L, Zavoli A, Colasurdo G (2019) A time-dependent TSP formulation for the design of an active debris removal mission using simulated annealing. In: AAS/AIAA Astrodynamics Specialist Conference, AAS 19-701, arXiv:1909.10427

Ferreira C, Figueira G, Amorim P (2022) Effective and interpretable dispatching rules for dynamic job shops via guided empirical learning. Omega 111:102643. https://doi.org/10.1016/j.omega.2022.102643

Gao S, Ruan Z, Floor P, et al (2026) Learning to solve orienteering problem with time windows and variable profits. In: International Conference on Learning Representations (ICLR), arXiv:2603.06260

Garza-Santisteban F, Cruz-Duarte JM, Amaya I, et al (2025) Selection hyper-heuristics and job shop scheduling problems: How does instance size influence performance? Journal of Scheduling 28:85–99. https://doi.org/10.1007/s10951-024-00819-8

Gunawan A, Lau HC, Vansteenwegen P (2016) Orienteering problem: A survey of recent variants, solution approaches and applications. European Journal of Operational Research 255:315–332. https://doi.org/10.1016/j.ejor.2016.04.059

Haeri Boroujeni SP, Razi A, Khoshdel S, et al (2024) A comprehensive survey of research towards AI-enabled unmanned aerial systems in pre-, active-, and post-wildfire management. Information Fusion 108:102369. https://doi.org/10.1016/j.inffus.2024.102369

Herrera EM, Panadero J, Carracedo P, et al (2022) Determining reliable solutions for the team orienteering problem with probabilistic delays. Mathematics 10(20):3788. https://doi.org/10.3390/math10203788

Holm S (1979) A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6(2):65–70

Hottung A, Tierney K (2022) Learning to solve a stochastic orienteering problem with time windows. Extended abstract, 11th Triennial Symposium on Transportation Analysis (TRISTAN XI)

Huang B, Cheng R, Li Z, et al (2025) EvoX: A distributed GPU-accelerated framework for scalable evolutionary computation. IEEE Transactions on Evolutionary Computation 29(5):1649–1662. https://doi.org/10.1109/TEVC.2024.3388550

Ingimundardottir H, Runarsson TP (2018) Discovering dispatching rules from data using imitation learning: A case study for the job-shop problem. Journal of Scheduling 21:413–428. https://doi.org/10.1007/s10951-017-0534-0

Jackson J, Mei Y (2020) Genetic programming hyper-heuristic with cluster awareness for stochastic team orienteering problem with time windows. In: IEEE Congress on Evolutionary Computation (CEC), https://doi.org/10.1109/CEC48606.2020.9185911

Jin H, Thomas BW (2019) Team orienteering with uncertain rewards and service times with an application to phlebotomist intrahospital routing. Networks 73(4):453–465. https://doi.org/10.1002/net.21875

Jones MW, Kelley DI, Burton CA, et al (2024) State of wildfires 2023–2024. Earth System Science Data 16:3601–3685. https://doi.org/10.5194/essd-16-3601-2024

Karunakaran D, Mei Y, Zhang M (2019) Multitasking genetic programming for stochastic team orienteering problem with time windows. In: IEEE Symposium Series on Computational Intelligence (SSCI), https://doi.org/10.1109/SSCI44817.2019.9002804

Kotthoff L (2014) Algorithm selection for combinatorial search problems: A survey. AI Magazine 35(3):48–60. https://doi.org/10.1609/aimag.v35i3.2460

Koza JR (1992) Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA

Krishnakumar K (1989) Micro-genetic algorithms for stationary and non-stationary function optimization. In: Proc. SPIE 1196, Intelligent Control and Adaptive Systems, pp 289–296, https://doi.org/10.1117/12.969927

Lan T, Srinivasa S, Wang H, et al (2022) WarpDrive: Extremely fast end-to-end deep multi-agent reinforcement learning on a GPU. Journal of Machine Learning Research 23(316):1–6

Lei Z, Hao JK, Wu Q (2025) Speeding up local optimization in vehicle routing with tensor-based GPU acceleration. arXiv preprint https://doi.org/10.48550/arXiv.2506.17357, arXiv:2506.17357

Li Y, Archetti C, Ljubić I (2024) Reinforcement learning approaches for the orienteering problem with stochastic and dynamic release dates. Transportation Science 58(5):1143–1165. https://doi.org/10.1287/trsc.2022.0366

Lim TY (2014) Structured population genetic algorithms: A literature survey. Artificial Intelligence Review 41:385–399. https://doi.org/10.1007/s10462-012-9314-6

Liu Y (2026) cuGenOpt: A GPU-accelerated general-purpose metaheuristic framework for combinatorial optimization. arXiv preprint https://doi.org/10.48550/arXiv.2603.19163, arXiv:2603.19163

Liu Y, Mei Y, Zhang M, et al (2017) Automated heuristic design using genetic programming hyper-heuristic for uncertain capacitated arc routing problem. In: Genetic and Evolutionary Computation Conference (GECCO), pp 290–297, https://doi.org/10.1145/3071178.3071185

López-Ibáñez M, Dubois-Lacoste J, Pérez Cáceres L, et al (2016) The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3:43–58. https://doi.org/10.1016/j.orp.2016.09.002

Marshall RJ, Johnston M, Zhang M (2014) Developing a hyper-heuristic using grammatical evolution and the capacitated vehicle routing problem. In: Simulated Evolution and Learning (SEAL), LNCS 8886, pp 668–679, https://doi.org/10.1007/978-3-319-13563-2_56

Mei Y, Zhang M (2018) Genetic programming hyper-heuristic for stochastic team orienteering problem with time windows. In: IEEE Congress on Evolutionary Computation (CEC), https://doi.org/10.1109/CEC.2018.8477983

Mei Y, Nguyen S, Xue B, et al (2017) An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming. IEEE Transactions on Emerging Topics in Computational Intelligence 1(5):339–353. https://doi.org/10.1109/TETCI.2017.2743758

Mendes A, França PM, Lyra C, et al (2005) Capacitor placement in large-sized radial distribution networks. IEE Proceedings - Generation, Transmission and Distribution 152(4):496–502. https://doi.org/10.1049/ip-gtd:20059015

Montemanni R, Gambardella LM (2009) An ant colony system for team orienteering problems with time windows. Foundations of Computing and Decision Sciences 34(4):287–306

Moscato P (1989) On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Tech. Rep. C3P Report 826, Caltech Concurrent Computation Program

Moscato P, Cotta C (2003) An introduction to memetic algorithms. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 7(19). https://doi.org/10.4114/ia.v7i19.721

Nguyen S, Zhang M, Johnston M, et al (2013) 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):621–639. https://doi.org/10.1109/TEVC.2012.2227326

Panadero J, Barrena E, Juan AA, et al (2022) The stochastic team orienteering problem with position-dependent rewards. Mathematics 10(16):2856. https://doi.org/10.3390/math10162856

Panadero J, Juan AA, Ghorbani E, et al (2024) Solving the stochastic team orienteering problem: comparing simheuristics with the sample average approximation method. International Transactions in Operational Research 31:3036–3060. https://doi.org/10.1111/itor.13302

Rice JR (1976) The algorithm selection problem. Advances in Computers 15:65–118. https://doi.org/10.1016/S0065-2458(08)60520-3

Sabar NR, Ayob M, Kendall G, et al (2013) Grammatical evolution hyper-heuristic for combinatorial optimization problems. IEEE Transactions on Evolutionary Computation 17(6):840–861. https://doi.org/10.1109/TEVC.2013.2281527

Secomandi N (2001) A rollout policy for the vehicle routing problem with stochastic demands. Operations Research 49(5):796–802. https://doi.org/10.1287/opre.49.5.796.10608

Sharifi I, Yildirim M, Fallah S (2025) Symbolic imitation learning: From black-box to explainable driving policies. arXiv preprint https://doi.org/10.3390/app152312464, arXiv:2309.16025

Shen S, Zhou Y, Lei Q, et al (2025) A survey of the orienteering problem: model evolution, algorithmic advances, and future directions. arXiv preprint https://doi.org/10.48550/arXiv.2512.16865, arXiv:2512.16865

Sim K, Hart E (2016) A combined generative and selective hyper-heuristic for the vehicle routing problem. In: Genetic and Evolutionary Computation Conference (GECCO), https://doi.org/10.1145/2908812.2908942

Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35(2):254–265. https://doi.org/10.1287/opre.35.2.254

Stone M (1974) Cross-validatory choice and assessment of statistical predictions. Journal of the Royal Statistical Society: Series B (Methodological) 36(2):111–147. https://doi.org/10.1111/j.2517-6161.1974.tb00994.x

Syswerda G (1989) Uniform crossover in genetic algorithms. In: Proceedings of the 3rd International Conference on Genetic Algorithms. Morgan Kaufmann, San Mateo,

CA, pp 2–9

Tesauro G, Galperin GR (1996) On-line policy improvement using Monte-Carlo search. In: Advances in Neural Information Processing Systems, vol 9. MIT Press, Cambridge, MA, pp 1068–1074

Toledo CFM, Oliveira L, França PM (2014) Global optimization using a genetic algorithm with hierarchically structured population. Journal of Computational and Applied Mathematics 261:341–351. https://doi.org/10.1016/j.cam.2013.11.008

Tsiligirides T (1984) Heuristic methods applied to orienteering. Journal of the Operational Research Society 35(9):797–809. https://doi.org/10.1057/jors.1984.162

Tyasnurita R, Özcan E, Drake JH, et al (2024) Constructing selection hyper-heuristics for open vehicle routing with time delay neural networks using multiple experts. Knowledge-Based Systems 295:111731. https://doi.org/10.1016/j.knosys.2024.111731

Vansteenwegen P, Souffriau W, Van Oudheusden D (2011) The orienteering problem: A survey. European Journal of Operational Research 209(1):1–10. https://doi.org/10.1016/j.ejor.2010.03.045

Varakantham P, Kumar A, Lau HC, et al (2018) Risk-sensitive stochastic orienteering problems for trip optimization in urban environments. ACM Transactions on Intelligent Systems and Technology 9(3). https://doi.org/10.1145/3080575

Vargha A, Delaney HD (2000) A critique and improvement of the CL common language effect size statistics of McGraw and Wong. Journal of Educational and Behavioral Statistics 25(2):101–132. https://doi.org/10.3102/10769986025002101

Wei J, Li Y, Zhao Y, et al (2026) Low-cost labels, reliable choices: Rollout-calibrated hyper-heuristics for job shop scheduling. arXiv preprint https://doi.org/10.48550/arXiv.2605.23957, arXiv:2605.23957

Wouda NA, Lan L, Kool W (2024) PyVRP: A high-performance VRP solver package. INFORMS Journal on Computing 36(4):943–955. https://doi.org/10.1287/ijoc.2023.0055

Wu Z, Wang L, Sun K, et al (2026) Enabling population-level parallelism in tree-based genetic programming for GPU acceleration. IEEE Transactions on Evolutionary Computation https://doi.org/10.1109/TEVC.2026.3663396

Xu L, Hutter F, Hoos HH, et al (2008) SATzilla: Portfolio-based algorithm selection for SAT. Journal of Artificial Intelligence Research 32:565–606. https://doi.org/10.1613/jair.2490

Yska D, Mei Y, Zhang M (2018) Feature construction in genetic programming hyper-heuristic for dynamic flexible job shop scheduling. In: Genetic and Evolutionary Computation Conference Companion (GECCO), pp 149–150, https://doi.org/10.1145/3205651.3205741

Yu VF, Salsabila NY, Lin SW, et al (2024) Simulated annealing with reinforcement learning for the set team orienteering problem with time windows. Expert Systems with Applications 238:121996. https://doi.org/10.1016/j.eswa.2023.121996

Zhang F, Mei Y, Nguyen S, et al (2021) Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job shop scheduling. IEEE Transactions on Cybernetics 51(4):1797–1811. https://doi.org/10.1109/TCYB.2020.3024849

Zhang F, Mei Y, Nguyen S, et al (2024) Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling. IEEE Transactions on Evolutionary Computation 28(1):147–167. https://doi.org/10.1109/TEVC.2023.3255246

Zhang S, Liu S, Lu N, et al (2025) LLM-driven instance-specific heuristic generation and selection. arXiv preprint arXiv:250600490 https://doi.org/10.48550/arXiv.2506.00490