Online Outperforms Offline: Reinforcement Learning Hyper-Heuristics for Vehicle Routing

Daniel R. Torres Ruiz[0000−0003−2640−2086], Daniel Karapetyan[0000−0003−4030−6525], Ender Ozcan[0000−0003−0276−1391], and Rong Qu[0000−0001−8318−7509]

University of Nottingham, Nottingham, UK

Keywords: Reinforcement Learning · Hyper-Heuristics · Delivery Planning

1 Introduction

Hyper-Heuristics (HHs) are a class of general-purpose methods that employ learning to reduce the cost of designing and implementing optimisation algorithms that are applicable in multiple problem domains. Selection HHs (SHHs) as a widely used type of HHs manages a set of domain-specific Low-Level Heuristics (LLHs) operators, such as mutation and hill-climbing. During the search process, SHH selects and applies these LLHs to explore the solution space.

SHHs can be divided into online and offline approaches. An online SHH learns the behaviour of LLHs and adapts the selection strategy during each run. Whereas an offline SHH follows a train-and-test paradigm: learning takes place on a training set of instances, and the learnt policy is then applied to the unseen instances. We experimentally compare these two approaches to show their strengths and weaknesses in practice. As an offline SHH, we employ Conditional Markov Chain Search (CMCS) that has been shown to produce effective search methods in several problem domains. CMCS is a single-point search method; at each iteration, it applies a selected LLH to the current solution, accepting any changes without backtracking. Choosing the next LLH depends on the previously applied LLH and its outcome, i.e., whether it improved the solution or not. The CMCS policy is learnt offline, remains the same during execution, and can potentially make the method particularly efficient under small time budgets.

Our online SHH is a new single-point search method based on Reinforcement Learning (RL). Like CMS, it also chooses the next LLH based on the last applied LLH and its outcome, accepting any change. However, its selection policy is learnt online using RL. As a result, it needs time to learn an effective policy; however, the policy adapts to the specific problem instance and the stage of the search, potentially making the method more effective under larger time budgets.

We compare these two approaches on the Vehicle Routing Problem with Time Windows (VRPTW) and show that our RL-based SHH indeed outperforms CMCS on larger time budgets, confirming the effectiveness of online learning.

2 Overview of methodology and related work

CMCS is a HH that selects the next LLH based on the outcome of the previously applied one through predefined transition configurations. These configurations are obtained via brute-force search, making the method sensitive to the number of LLHs as well as the size and representativeness of the training set. As a result, CMCS relies on fixed configurations whose performance may degrade when the training and test distributions differ, often requiring retuning and limiting its use in dynamic settings. CMCS has been evaluated on the Uncapacitated Facility Location [2] and Bipartite Boolean Quadratic Programming [1] problems, but no prior work evaluates it on the VRPTW in comparison with other HHs .

Selection HHs (SHHs) typically manage a set of perturbative LLHs and iteratively apply a selected LLHs to improve a solution, employing a high-level heuristic selection strategy combined with a move acceptance criterion [3]. The HyFlex framework enables the development and evaluation of general-purpose HHs by providing a standardised, problem-independent interface that encapsulates solution representation, initialisation, evaluation, and LLHs.

This general-purpose setting exposes a limitation of CMCS. HyFlex supports adaptable methods for multiple problem domains, whereas CMCS relies on offline configurations tuned for a specific domain. This creates a generality gap between general-purpose approaches and offline fixed policies tailored to particular problem characteristics. Bridging this gap requires adaptive selection mechanisms that operate online without pre-tuning and guide the decision-making process.

RL introduces such adaptivity in SHHs, but existing approaches remain limited. The RL Great Deluge HH (RLGDHH, [4]) assigns utility values to LLHs using a simple fixed reward-punishment strategy. It does not use Q-learning or a state representation; instead, it updates heuristic utilities solely based on the objective values of the solutions produced. As a result, its policy is context-independent and cannot capture interactions between heuristics during the search. This limitation motivates adopting a minimal state representation, as in CMCS, using Q-learning with TD(0) updates to guide the search process.

3 Preliminary experiments

Evaluation is conducted on all RC instances from Solomon and Homberger, with multiple seeded runs per instance and a 600-second time budget. The percentage gap metric is used and the results are compared statistically.

The configurations of the CMCS transition matrix are obtained following [1], using a broad set of instances and ten available LLHs. The best-performing configurations with 2 and 3 LLHs define the baselines CMCS-2 and CMCS-3. To assess the state representation, we include two additional variants: O_2, with only the previous LLH, and a stateless bandit-based approach.

3.1 Results

Experimental results show that online approaches consistently outperform offline fixed policies during the search. Figure 1a shows the mean gap over time for Solomon instances. All RL-based methods outperform CMCS-2 and CMCS-3. RLGDHH converges more slowly but reaches competitive performance after 500 seconds. CMCS-3 improves over CMCS-2 across the full time budget, indicating that adding a third operator improves exploration and helps avoid local minima.

Figure 1b shows similar trends on Homberger instances, although online methods require more time to surpass offline approaches due to the increased problem size. In contrast to Solomon, RLGDHH performs better than O_3 and its variants in early steps. CMCS-3 does not improve over CMCS-2, suggesting that the additional operator introduces overhead that is not compensated in larger instances. As in Figure 1c for both benchmarks, the final gap distributions reflect this CMCS-3 effect, particularly on high-demand RC Homberger instances.

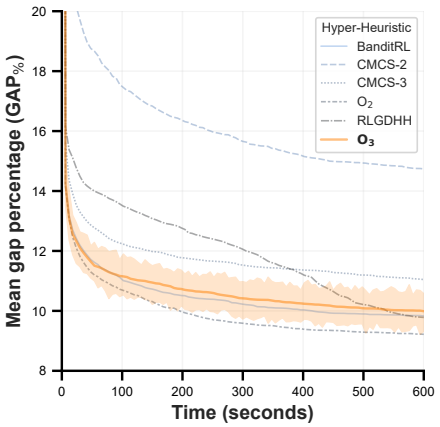
The computational cost explains this particular behaviour. In Figure 1d, online methods make fewer heuristic calls per second than offline approaches, except for the stateless variant on Solomon. The additional operator in CMCS-3 increases evaluation cost and limit the search, highlighting a trade-off between exploration and search throughput, given the instance size. Despite its simplicity, CMCS-2 remains competitive with only a mutation and a local search heuristic.

Pairwise Wilcoxon tests indicate significant differences between offline and online approaches ($p < 0.05$), but no differences among the online RL approaches ($p \geq 0.05$). Despite differing state representations, all RL methods reach similar final performance. However, their search behaviour differs from RLGDHH (Figure 1e). Crossover operators dominate across different state variants, indicating that combining the current and best solutions is the main driver of performance.

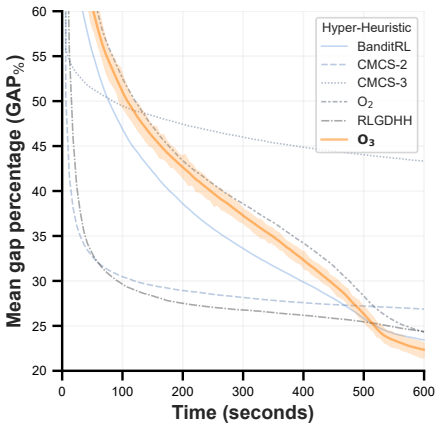
Online HHs consistently outperform offline configurations across both benchmarks, despite using fewer calls. Adaptive selection compensates for reduced search throughput by improving decision quality, highlighting the robustness of online learning and the offline policies limitations when problem scale increases.

References

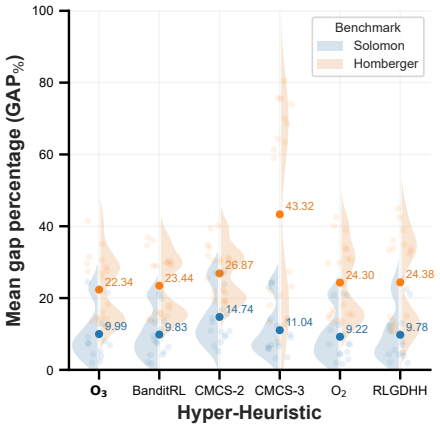
1. Karapetyan, D., Goldengorin, B.: Conditional markov chain search for the simple plant location problem improves upper bounds on twelve körkel–ghosh instances. In: Optimization Problems in Graph Theory, vol. 139, pp. 123–147. Springer International Publishing (2018). https://doi.org/10.1007/978-3-319-94830-0_7
2. Karapetyan, D., Punnen, A.P., Parkes, A.J.: Markov chain methods for the bipartite boolean quadratic programming problem. European Journal of Operational Research **260**(2), 494–506 (2017). https://doi.org/10.1016/j.ejor.2017.01.001
3. Li, C., Wei, X., Wang, J., Wang, S., Zhang, S.: A review of reinforcement learning based hyper-heuristics. PeerJ Computer Science **10**, e2141 (2024). https://doi.org/10.7717/peerj-cs.2141
4. Ozcan, E., Misir, M., Ochoa, G., Burke, E.K.: A Reinforcement Learning - Great-Deluge Hyper-Heuristic for Examination Timetabling. International Journal of Applied Metaheuristic Computing **1**(1), 39–59 (2010). https://doi.org/10.4018/jamc.2010102603



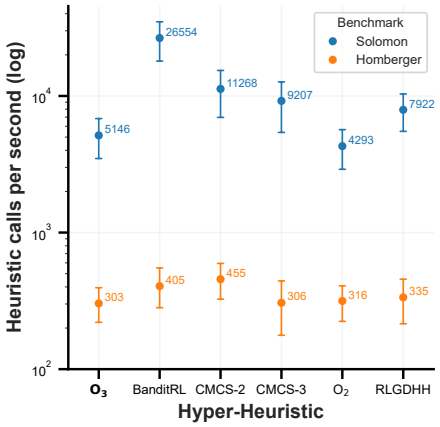
(a) Solomon (mean gap over time)



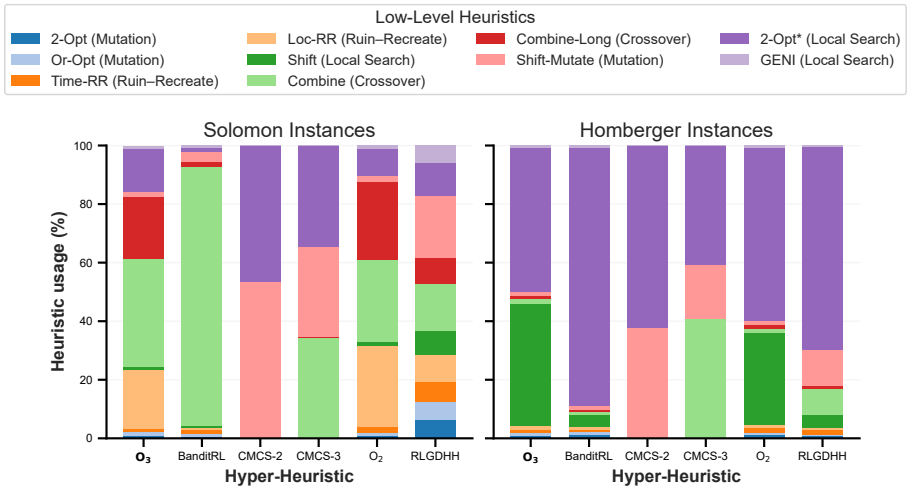
(b) Homberger (mean gap over time)



(c) Gap distributions



(d) Calls per second



(e) Usage composition

Fig. 1: Convergence profiles on (a) Solomon and (b) Homberger; (c) final percentage gap distributions; (d) heuristic calls per second; and, (e) LLH usage.