Particle Swarm Optimization Configures the Route Minimization Algorithm*

Tomasz Jastrzab[1][0000−0002−7854−9058], Michal Myller[1][0000−0001−9265−3997], Lukasz Tulczyjew[1][0000−0003−0763−0745], Miroslaw Blocho[2], Wojciech Ryczko[2], Michal Kawulok[1][0000−0002−3669−5110], and Jakub Nalepa[1][0000−0002−4026−1569]

[1] Department of Algorithmics and Software, Silesian University of Technology, Gliwice, Poland
[2] Blees, Gliwice, Poland
jnalepa@ieee.org

Abstract. Solving rich vehicle routing problems is an important topic due to their numerous practical applications. Although there exist a plethora of (meta)heuristics to tackle this task, they are often heavily parameterized, and improperly tuned hyper-parameters adversely affect their performance. We exploit particle swarm optimization to select the pivotal hyper-parameters of a route minimization algorithm applied to the pickup and delivery problem with time windows. The experiments, performed on benchmark and real-life data, show that our approach automatically determines high-quality hyper-parameters of the underlying algorithm that improve its abilities and accelerate the convergence.

Keywords: Vehicle routing problem · route minimization · hyper-parameter selection · particle swarm optimization · PDPTW.

1 Introduction

Solving rich vehicle routing problems (VRPs) is an important research topic, as their practical applications span across different domains [12]. There are a multitude of various VRP formulations that reflect real-life constraints, including limited vehicle capacities, time windows, or the maximum waiting times [15]. Commonly, the main objective is to minimize the number of vehicles (K) serving all requests, whereas the secondary objective is to optimize the distance (T) traveled in a routing schedule. Since VRPs are often NP-hard, the most widely-used algorithms to approach such discrete optimization problems include various (meta)heuristics—they do not ensure obtaining optimal solutions, but work much faster than exact techniques and can be utilized in large-scale scheduling.

Although heuristics for VRPs were shown efficient in virtually all VRPs, they are usually heavily parameterized, and the incorrect hyper-parameters lead to

* This work was supported by the European Union funds awarded to Blees Sp. z o. o. under grants POIR.04.01.01-00-0079/18-01 and UDA-RPSL.01.02.00-24-00FG/19-00. JN was supported by the Silesian University of Technology grant for maintaining and developing research potential.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08754-7_11

sub-optimal solutions. We tackle this issue and exploit particle swarm optimization (PSO) to automate the process of selecting the pivotal hyper-parameters of the route minimization algorithm (Section 2) applied to the pickup and delivery problem with time windows (PDPTW). It is formally defined in Section 1.1, whereas Section 1.2 presents the current state of the art in solving this VRP variant. Our experiments, performed over benchmark and real-life data, showed that PSO consistently evolves high-quality hyper-parameters that boost the abilities of the optimization technique and accelerate its convergence (Section 3).

1.1 Problem Formulation

The PDPTW can be defined on a directed graph $G = (V, E)$, with a set V of $B + 1$ vertices and a set of edges E. The vertices v_i, $i \in \{1, ..., B\}$, represent the locations of the requests, and v_0 indicates the depot. A set of edges $E = \{(v_i, v_{i+1}) | v_i, v_{i+1} \in V, v_i \neq v_{i+1}\}$ represents the links between particular customers. The costs $c_{i,j}$, $i, j \in \{0, 1, ..., C\}$, $i \neq j$, are equal to the distances between the travel points. Each request z_i, $i \in \{1, ..., n\}$, where $n = B/2$, is a coupled pair of pickup (P) and delivery (D) customers indicated by p_z and d_z, respectively, where $P \cup D = V \setminus \{v_0\}$ and $P \cap D = \emptyset$. For each request z_i, the amount of delivered ($q^d(z_i)$) and picked up ($q^p(z_i)$) demand is defined, where $q^d(z_i) = -q^p(z_i)$. Each customer v_i defines its demand (pickup or delivery), service time s_i ($s_0 = 0$ for the depot), and time window $[e_i, f_i]$ within which either pickup or delivery service should start (it can be completed after f_i).

In PDPTW, the fleet is homogeneous (with K denoting its size), the capacity of each vehicle is constant (Q). Each route r in the solution σ (which is a set of routes), starts and finishes at the depot. In the PDPTW, minimizing the fleet size is the primary objective, and decreasing the distance is the secondary one.

1.2 Related Work

The algorithms for minimizing the number of routes in the PDPTW include exact and approximate methods. The former techniques can find an optimal solution [11], but their main downside is that they are only applicable to small and moderate-sized problem instances [6]. On the other hand, the approximate methods allow us to find near-optimal solutions to larger problems in a short time. The heuristics fall into one of the three categories: construction, two-phase, and improvement algorithms [4]. Construction algorithms build the solution from scratch by seeding new routes when necessary. When none of the vehicles can handle a request[3], a new one is taken from the fleet. In [8], the authors seed a new route only after both *insertion* and *exchange* operations fail to introduce the request to the existing routes. Two-phase algorithms follow two general schemes: cluster-first route-second, and route-first cluster-second. The main idea is to reduce the search space by combining multiple customers into clusters. Zunic et. al [16] split the set of customers into clusters based on the distance to the depot.

[3] The reasons for this inability may be capacity or time window constraint violation.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08754-7_11

Combining vehicles from different clusters is not allowed, but adding unclustered customers to a particular cluster is possible. Moreover, the proposed algorithm allows for hiring vehicles (if necessary) at a higher cost. The improvement algorithms start with a low-quality solution and minimize the fleet size and distance. The improvements result from relocations or exchanges of route fragments [7].

The metaheuristics involve population-based and local search approaches [3]. The first group focuses on bio-inspired techniques such as ant or bee colony optimizations, particle swarm optimization, firefly or bat algorithms [1,14], and the genetic and memetic ones [2]. The local search algorithms include tabu searches, simulated annealing, or the greedy search procedures, as well as a variety of neighborhood search methods [1,5]. Their main drawback is that they are heavily parameterized, and the parameter tuning process is time-consuming. Therefore, some techniques employ run-time adaptation through e.g., analyzing the characteristics of the problem instance and selecting the best parameters for the given problem variant. A different approach is proposed in [17], where the tuning exploits the historical data collected through GPS devices. Overall, we follow this research pathway and introduce PSO for optimizing the pivotal parameters of a guided ejection search-powered route minimization algorithm for the PDPTW [13]. This algorithm has been shown to be outperforming other approaches for minimizing the number of routes in the PDPTW, therefore we focus on it here. However, PSO is independent from the underlying route minimization technique, thus can be effectively deployed to optimize other algorithms too. Additionally, this metaheuristics is utilized because it offers high-quality performance in other tasks that involve hyper-parameters' tuning [9].

2 Configuring the Guided Ejection Search Using PSO

At each step of the guided ejection search (GES, Algorithm 1; we also report the time complexity of each step[4], where n is the number of transportation requests), inspired by [13], a random route r is drawn, its requests are inserted into the *ejection pool* (EP) (line 3), and then up to ϵ attempts to re-insert them back into the solution σ are undertaken. The *tabu pool* (TP) of maximum size α and with the maximum number of occurrences of the same entry β is initiated (line 4). The penalty counters (PCs) p indicate the re-insertion difficulty of the request. A request h_{in} is popped from the EP (line 8). If there are several feasible positions to insert h_{in} into σ, a random one is chosen (line 10). Otherwise, the inserted request violates the constraints, and the solution with h_{in} is squeezed through local moves to restore its feasibility (line 11). If squeezing fails, the request's p is increased (line 14), and h_{out} (with minimal $p[h_{out}]$) is ejected to insert h_{in} (lines 15–19). Finally, σ is mutated with a probability π using out/in-relocate and out/in-exchange moves (line 20)—the maximum and feasible number of mutation moves cannot exceed κ and λ, respectively. The optimization finishes once the maximum time has elapsed (in this work, it is two minutes).

[4] Although for *Squeeze* and *Mutate*, their time complexity is fairly high, it is their worst-case complexity, and these procedures terminate much faster in practice.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08754-7_11

Algorithm 1 The route minimization guided ejection search.

$\sigma_{best} \leftarrow \sigma_{init}$; $\sigma \leftarrow \sigma_{init}$ ▷ $\theta(n)$
while not *stop condition* **do**
  Initialize EP with requests from a random r ▷ $O(n)$
  Initialize tabu pool TP(α, β) ▷ $O(1)$
  Initialize PCs as $p[h_j] := 1, j = 1, 2, ..., n$ ▷ $\theta(n)$
  Initialize iteration counter $k := 1$ ▷ $O(1)$
  while EP $\neq \emptyset$ **and** $k \leq \epsilon$ **do** ▷ max. ϵ iterations
    $k \leftarrow k + 1$; Select and remove h_{in} from EP ▷ $O(1)$
    if $S^{fe}_{in}(h_{in}, \sigma) \neq \emptyset$ **then** ▷ $O(n^2)$
      $\sigma \leftarrow$ random $\sigma' \in S^{fe}_{in}(h_{in}, \sigma)$ ▷ $O(1)$
    else $\sigma \leftarrow$ Squeeze(h_{in}, σ) ▷ $O(n^4)$
    end if
    if $h_{in} \notin \sigma$ **then** ▷ $O(1)$
      $p[h_{in}] := p[h_{in}] + 1$ ▷ $O(1)$
      Generate $S^{fe}_{ej}(h_{in}, \sigma)$ with min $p[h_{out}]$ ▷ $O(n^3)$
      if $S^{fe}_{ej}(h_{in}, \sigma) \neq \emptyset$ **then** ▷ $O(1)$
        $\sigma \leftarrow$ random $\sigma' \in S^{fe}_{ej}(h_{in}, \sigma)$ ▷ $O(1)$
        Add the ejected request h_{out} to EP ▷ $O(1)$
      end if
      $\sigma \leftarrow$ Mutate(σ, π, λ, κ) ▷ $O(\lambda n^4)$
    end if
  end while
  if EP $\neq \emptyset$ **then** $\sigma \leftarrow \sigma_{best}$ ▷ $\theta(n)$
  else $\sigma_{best} \leftarrow \sigma$ ▷ $\theta(n)$
  end if
end while
return best solution σ_{best} ▷ $O(1)$

In PSO [9], each particle's position encodes m hyper-parameters (Table 1, also in blue in Algorithm 1), and we maximize the fitness[5]: $\mathcal{F} = (0.5 \cdot (K - K_B)/K_B + 0.5 \cdot \tau_B/\tau)^{-1}$, where K and τ are the number of routes and the maximum execution time of GES with the corresponding hyper-parameters, and K_B and τ_B are the best-known number of routes for the corresponding instance in the validation set $\boldsymbol{V}$, and the best time required to converge to the solution with K routes captured during the evolution. This fitness function allows us to capture both functional and non-functional aspects of the algorithm (here, in relation to the best-known routing schedules). We evolve s random particles sampled from a uniform distribution bounded by the lower and upper parameter limits with zero initial velocity updated in each iteration for the *i-th* particle: $\boldsymbol{v}_i \leftarrow \omega \boldsymbol{v}_i + \phi_p \boldsymbol{r_p}(\boldsymbol{\lambda}^*_i - \boldsymbol{\lambda}_i) + \phi_g \boldsymbol{r_g}(\boldsymbol{\lambda}^S - \boldsymbol{\lambda}_i)$, where $\boldsymbol{r_p}, \boldsymbol{r_g}$ are from a uniform distribution $\mathcal{U}(0, 1)$, ω is the inertia, ϕ_p, ϕ_g are the acceleration coefficients, and $\boldsymbol{\lambda}^*_i$ and $\boldsymbol{\lambda}^S$ are the best *i-th* particle's and best swarm's positions visited. The *i-th* particle's position becomes $\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_i + \boldsymbol{v}_i$. Once the evolution is finished, we pick the best particle that corresponds to the highest-quality parameterization.

3 Experiments

We use six Li and Lim's tests, one for the instances with clustered (c), randomized (r), and mixed (rc) requests, with short time windows and small vehicle

[5] In PSO, the fitness function can be updated to reflect other aspects of the solutions.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08754-7_11

capacities (c1, r1, and rc1), and with wider windows and larger vehicles (c2, r2, rc2). We focus on 200-request tests[6], and take the first problem instance from each group to form $\boldsymbol{V}$. The max. time of calculating a fitness of a single particle amounts to 12 minutes (as the max. time for GES is 2 min). We pick the best hyper-parameters from each configuration for each run, and apply them to GES to solve all Li and Lim's tests (60 instances), and our 40 real-life tests of various characteristics[7]. We used the Python PSO with default parameters [9], and GES was coded in C#, and ran on Intel Xeon CPU E5-2640 v3, 2.60 Ghz, 8 GB RAM.

Table 1. The hyper-parameters of the guided ejection search.

Symbol	Range	Step	Expert's	PSO	Meaning
α	[1, 10]	1	8	3	maximum tabu pool size
β	[2, 4]	1	2	3	max. number of occurrences of the same entry in TP
ϵ	[1, 1000]	1	50	616	maximum number of request ejections
π	[0.0, 1.0]	Cont.	0.25	0.41	mutation probability
λ	[1, 200]	1	100	2	feasible number of mutations
κ	[1, 10000]	1	1000	6062	maximum number of mutations

We keep the number of fitness evaluations ($s \cdot G_{\max}$), where $G_{\max}$ is the number of generations, constant. We investigate the impact of the swarm size s and maximum number of generations on PSO, and consider the $(s, G_{\max})$ pairs: $(12, 2)$, $(8, 3)$, $(6, 4)$, $(4, 6)$, $(3, 8)$, and $(2, 12)$—each pair was run seven times. We confront PSO with Expert's (the hyper-parameters determined by an expert) and random search with 24 uniformly distributed sample points to ensure fair comparison. We always seed the *same initial solution* σ_{init} for all GES variants.

Table 2. The (a) best fitness (the best mean and median are in bold) and (b) distance traveled by the best particle in PSO.

PSO sett.→		(12, 2)	(8, 3)	(6, 4)	(4, 6)	(3, 8)	(2, 12)
(a)	mean	0.487	0.490	0.476	**0.498**	0.481	0.451
	std dev.	0.019	0.015	0.012	0.012	0.022	0.029
	median	0.487	0.492	0.478	**0.502**	0.475	0.458
(b)	mean	0.070	0.142	0.149	0.300	0.266	0.167
	std dev.	0.024	0.060	0.041	0.059	0.168	0.166
	median	0.078	0.159	0.174	0.317	0.243	0.082

Table 2 gathers the PSO results aggregated for all $(s, G_{\max})$ pairs. We can observe that balancing the swarm size and the maximum number of generations in $(4, 6)$ leads to the largest fitness values obtained over $\boldsymbol{V}$, hence the evolutionary process guides the particles toward high-quality parts of the solution space. The exploration capabilities of PSO are reflected in the distance traveled by the best

[6] `https://www.sintef.no/projectweb/top/pdptw/li-lim-benchmark/200-customers/`

[7] This set and the baseline solutions are available at `https://gitlab.com/tjastrzab/iccs2022/`.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08754-7_11

particle in the swarm: very small swarms of two particles tend to travel larger distances, whereas larger ones, (12, 2), exploit the search space more locally, as they likely captured well-fitted random individuals in the initial population.

In Fig. 1, we confront PSO with the manual tuning process (Expert's) and random sampling. Here, we present the parameterizations that lay on the Pareto fronts, hence are not dominated by other solutions—the closer the solutions are to the point $(0, 0)$, the better. PSO consistently delivers hyper-parameters that lead to faster convergence compared with those selected by a human expert, while maintaining low K's—the differences in K for Expert's and $(4, 6)$ are not statistically significant (Friedman's test with Dunn's, $p < 0.05$). In Table 1, we presented the best values extracted by PSO (it was delivered by the $(4, 6)$ configuration with the fitness of 0.52)—they are significantly different from Expert's.

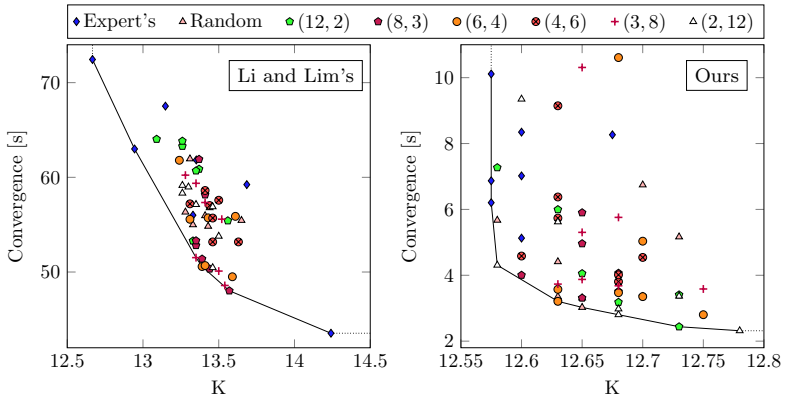

Fig. 1. The Pareto fronts (K vs. convergence in seconds, averaged across all instances in the corresponding dataset) obtained for the Li and Lim's and our test sets.

4 Conclusions

In this paper, we tackled the problem of automated parameterization of the algorithms solving rich vehicle routing problems, and employed PSO to evolve the pivotal hyper-parameters of a heuristics for minimizing routes in the PDPTW. The experiments showed that it elaborates high-quality hyper-parameters working on par with the GES parameterization delivered by a human expert, while allowing for faster convergence. Our current research efforts are focused on integrating PSO with the famous iterated racing procedures available in irace [10]. We believe that developing the solvers with the automated process of retrieving their desired parameterizations is an important step toward data-driven algorithms that will be able to solve emerging formulations of rich VRPs.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08754-7_11

References

1. Blocho, M.: Heuristics, metaheuristics, and hyperheuristics for rich vehicle routing problems. In: Nalepa, J. (ed.) Smart Delivery Systems. Solving Complex Vehicle Routing Problems, pp. 101–156. Intelligent Data Centric Systems, Elsevier (2020)
2. Blocho, M., Nalepa, J.: LCS-based selective route exchange crossover for the pickup and delivery problem with time windows. In: Evolutionary Computation in Combinatorial Optimization. LNCS, vol. 10197, pp. 124–140. Springer (2017)
3. Feng, L., Zhou, L., Gupta, A., Zhong, J., Zhu, Z., Tan, K.C., Qin, K.: Solving generalized vehicle routing problem with occasional drivers via evolutionary multitasking. IEEE Transactions on Cybernetics pp. 1–14 (2019)
4. Konstantakopoulos, G., Gayialis, S., Kechagias, E.: Vehicle routing problem and related algorithms for logistics distribution: a literature review and classification. Operational Research (2020)
5. Lai, D., Demirag, O., Leung, J.: A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph. Transp. Res. Part E **86**, 32–52 (2016)
6. Lee, C.: An exact algorithm for the electric-vehicle routing problem with nonlinear charging time. Journal of the Operational Res. Society **72**(7), 1461–1485 (2021)
7. Li, H., Li, Z., Cao, L., Wang, R., Ren, M.: Research on optimization of electric vehicle routing problem with time window. IEEE Access **8** (2020)
8. Liu, J., Feng, S., Niu, Q., Li, L.: New construction heuristic algorithm for solving the vehicle routing problem with time windows. IET Collaborative Intelligent Manufacturing **1**, 90–96 (2019)
9. Lorenzo, P.R., Nalepa, J., Kawulok, M., Ramos, L.S., Pastor, J.R.: Particle swarm optimization for hyper-parameter selection in deep neural networks. In: Proc. GECCO. p. 481–488. ACM, USA (2017)
10. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43–58 (2016)
11. Mohamed, E., Ndiaye, M.: Optimal routing and scheduling in e-commerce logistics using crowdsourcing strategies. In: Proc. IEEE ICITM. pp. 248–253 (2018)
12. Mor, A., Speranza, M.G.: Vehicle routing problems over time: a survey. 4OR **18**(2), 129–149 (2020)
13. Nalepa, J., Blocho, M.: Adaptive guided ejection search for pickup and delivery with time windows. Journal of Intelligent & Fuzzy Systems **32**, 1547–1559 (2017)
14. Osaba, E., Yang, X., Fister Jr., I., Del Ser, J., Lopez-Garcia, P., Vazquez-Pardavila, A.: A discrete and improved bat algorithm for solving a medical goods distribution problem with pharmacological waste collection. Swarm and Evolutionary Computation **44**, 273–286 (2019)
15. Zhang, H., Ge, H., Yang, J., Tong, Y.: Review of vehicle routing problems: Models, classification and solving algorithms. Archives of Computational Methods in Engineering **29**(1), 195–221 (2022)
16. Zunic, E., Donko, D., Supic, H., Delalic, S.: Cluster-based approach for successful solving real-world vehicle routing problems. In: Proc. ACSIS. vol. 21, pp. 619–626. Springer (2020)
17. Žunić, E., Delalić, S., Donko, D.: Adaptive multi-phase approach for solving the realistic vehicle routing problems in logistics with innovative comparison method for evaluation based on real GPS data. Transportation Letters **14**(2), 143–156 (2022)

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08754-7_11