FILO2^{xe} for Galileo: Linking Multiple FILO2^{x} Extended Runs

Luca Accorsi[1], Demetrio Laganà[2], Federico Michelotto[3], Roberto Musmanno[2], and Daniele Vigo[3,4]

[1] Google
[2] DIMEG, University of Calabria, Rende (CS), Italy
[3] DEI "G. Marconi", University of Bologna, Bologna, Italy
[4] CIRI-ICT, University of Bologna, Bologna, Italy

Abstract. This paper describes the extension of the FILO2^{x} algorithm, a parallel metaheuristic for the Capacitated Vehicle Routing Problem, designed to perform extremely long runs as required by the CVRPLib Best Known Solution Challenge. The approach was used by Team Galileo to participate to the challenge. The team overall ranked 6th and obtained several best known solutions (BKSs), three of which resisted until the end of the challenge.

Keywords: Capacitated Vehicle Routing Problem · Heuristic Algorithms · Parallel Algorithms

1 Introduction

The Capacitated Vehicle Routing Problem (CVRP) is one of the most studied combinatorial optimization problems, which calls for the determination of the set of routes used to serve a set of customers, each requiring the delivery of known quantities of goods, by using a fleet of vehicles with limited capacity. Since it was first introduced by Dantzig and Ramser [9], a large number of exact and heuristic methods were proposed for the solution of the CVRP (see Toth and Vigo [13] and [14]). Recently, the literature on heuristics focused on the development of methods capable of solving large-scale instances with several thousands of customers (see Arnold et al. [7] and Accorsi and Vigo [4]).

To promote research on the solution large-scale CVRP instances, the CVRPLib [1] team proposed a new benchmark set of 100 instances with up to 10,000 customers and organized an international challenge to obtain high quality solutions for these instances [2].

Participants were invited to submit feasible solutions that improve the initial BKSs established by the organizers. These initial benchmarks were established by running state-of-the-art algorithms 60 times for every instance, using a different random number generator seed each time, and a two-hour time limit [3]. Notably, the initial BKSs have been found by using just three methods: AILS-II

[12] (which obtained 93 out of 100 initial BKSs), FILO2 [4] (6 out of 100), and FILO [6] (1 out of 100).

2 Background: FILO2^x

FILO2^x is a single-trajectory parallel metaheuristic for efficiently solving large-scale CVRP instances (see [5]).

The core optimization procedure, based on the iterated local search paradigm, proposed by Lourenço et al. [11], is the primary tool for improving initial solutions. These solutions are generated using an adaptation of the Savings algorithm proposed by Clarke and Wright [8], followed by an optional quick route minimization procedure.

Each core optimization iteration generates a neighbor solution S' by applying ruin, recreate, and local search steps to a current reference solution S. The neighbor solution S' may replace S according to a standard simulated annealing (SA) acceptance criterion (see Kirkpatrick et al. [10]).

The difference between S and S' is usually small, since optimization steps are extremely localized. A number of x solvers are then used to perform multiple core optimization iterations in parallel. This results in a low probability of generating overlapping changes that lead to infeasible neighbor solutions. Iterations that result in infeasibilities are discarded.

The core optimization procedure is performed for Δ_{CO} iterations, during which the SA temperature is exponentially lowered from t_0 to t_f.

3 Extension of FILO2^x for the BKS Challenge

The solution approach proposed for the BKS challenge, hereafter referred to as FILO2^{xe}, builds on the FILO2^x approach and extends it to handle long-running and potentially indefinite executions.

FILO2^{xe} performs several rounds of the core optimization procedure. During the first round, as in FILO2^x, Δ_{CO} iterations are performed while the SA temperature is lowered from t_0 to t_f. Subsequent rounds perform $\bar{\Delta}_{CO}$ iterations with an initial simulated annealing temperature equal to $\bar{t}_0$ and the standard final temperature t_f, where $\bar{\Delta}_{CO}$ and $\bar{t}_0$ are randomly selected from uniform distributions in $[0.1 \cdot \Delta_{CO}, 10 \cdot \Delta_{CO}]$ and $[\hat{t}_0, \hat{t}_1]$, respectively. Initially, $\hat{t}_0$ is defined as $1.5 \cdot t_0$ and $\hat{t}_1$ is set to $1.5 \cdot \hat{t}_0$. At every restart, these temperatures are updated to $\hat{t}_0 \leftarrow \hat{t}_1$ and $\hat{t}_1 \leftarrow min\{1.5 \cdot \hat{t}_1, t_f\}$. Whenever $\hat{t}_0$ equals t_f, the temperatures are re-initialized to $\hat{t}_0 \leftarrow 1.5 \cdot t_0$ and $\hat{t}_1 \leftarrow 1.5 \cdot \hat{t}_0$. The cooling schedule is adjusted to match the new temperature range and number of iterations.

The first round starts with the initial solution obtained by the savings and route minimization procedures. Subsequent rounds are restarted with a previously

visited solution found at temperature $\bar{t}_0$. Specifically, the algorithm's search trajectory is discretized into k buckets of exponentially decreasing size based on the initial SA temperatures t_0 and t_1. Each bucket b, representing a temperature range $[t_a, t_b]$, contains a solution found within that range. In particular, a solution found at temperature $t \in [t_a, t_b]$ can replace the current solution stored in the associated bucket if accepted with a SA-based criterion.

Finally, every solution that improves the best solution in the current round is sent to a solver exclusively performing intensification procedures. More specifically, the solver keeps the 10 best solutions found so far and at each iteration randomly selects one of them, performs the ruin, recreate, and local search steps, and only accepts the resulting solution if it is improving with respect to the starting ones.

For some instances with many routes, a parallel iterative route-based decomposition approach was evaluated, named FILO2^{y}. Starting from an initial solution computed as in FILO2, this approach partitions the current solution into disjoint subsets of routes, creating as many sub-problems as the partition cardinality. Each sub-problem is then optimized using FILO2 for a prescribed number of core optimization iterations, and the resulting solutions are then merged to form the new current solution. The simulated annealing temperature in each sub-problem is defined to match the cooling schedule of the sequential algorithm FILO2. The process is iterated until the final temperature t_f is reached, at which point the temperature is restarted with a randomly selected value from $[t_f/10, t_0/2]$.

4 Computational Setting

The algorithm was executed on 100 AWS shared cloud machines (c6g.xlarge), equipped with four virtual processors and 8 GB of RAM.

Initially, each instance was optimized on a distinct virtual machine. Subsequently, the exact number of machines was determined based on forecast of cloud expenditures and respecting a maximum budget of $10000.

Occasionally, we also used an additional 64-bit GNU/Linux Ubuntu 22.04 workstation with an Intel Xeon Gold 6254 CPU (3.1 GHz) and 384 GB of RAM.

5 Algorithm Parameters

All parameters are inherited from FILO2^{x} (see Accorsi et al. [5]). The initial number of core optimization iterations Δ_{CO} is set to one million, consistent with long runs of FILO2^{x}. As a consequence, $\bar{\Delta}_{CO}$ ranges from one hundred thousand to ten million. The simulated annealing temperatures t_0 and t_f are defined to be proportional to the average cost of an arc in an instance.

Four solvers ($x = 4$) have been used for experiments on cloud machines, whereas up to 16 solvers have been used to process selected instances on the additional

workstation. Finally, an additional thread has been used to perform the intensification.

6 Achieved Results

The testing has been performed on the 100 XL instances with 1,000 to 10,000 customers produced for the challenge. A total solution time of one month was available and team Galileo, which employed the described approach, overall ranked 6th, obtaining several BKS, three of which resisted till the end of the challenge (two obtained with FILO^{xe} and one with FILO^{y}). We note that out of the 16 registered teams for the challenge only seven were able to obtain at least one BKS. In the following table, we report the final results of the challenge, where for each team we list the final score and the number of BKS at the end of the challenge. Furthermore, for Team Galileo, which is the only one we are aware of the detailed results on each instance, the average percentage gap of the best solution found, z, with respect to the BKS, defined as $(z - BKS)/BKS * 100$, is equal to 0.117%. Such gap is in line if not better than the typical gaps achieved by FILO and FILO2 on other benchmarks from the literature.

Table 1. Final results of the BKS challenge

Rank	Team	Overall Score	# BKS
1	OptVerse-CityU	1,800.31632	51
2	TQrouting	904.11095	27
3	Sub-Appro	267.49942	6
4	AILSII+	156.15780	8
5	AILS-HGS	103.62808	4
6	Galileo	75.99384	3
7	Sorry I'm Late	0.25263	0

References

1. CVRPLib (Jan 2026), https://galgos.inf.puc-rio.br/cvrplib, last accessed on Feb 25, 2026
2. CVRPLib Best Known Solution Challenge (Jan 2026), https://galgos.inf.puc-rio.br/cvrplib/index.php/en/bks_challenge/overview, last accessed on Feb 25, 2026
3. The XL instances for the capacitated vehicle routing problem (Jan 2026), https://galgos.inf.puc-rio.br/cvrplib/uploads/files/report.pdf, last accessed on Feb 25, 2026
4. Accorsi, L., Vigo, D.: Routing one million customers in a handful of minutes. Computers & Operations Research **164**, 106562 (2024). https://doi.org/10.1016/j.cor.2024.106562
5. Accorsi, L., Laganà, D., Michelotto, F., Musmanno, R., Vigo, D.: Asynchronous cooperative optimization of a capacitated vehicle routing problem solution (2025), https://arxiv.org/abs/2511.19445

6. Accorsi, L., Vigo, D.: A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems. Transportation Science **55**(4), 832–856 (2021). https://doi.org/10.1287/trsc.2021.1059
7. Arnold, F., Gendreau, M., Sörensen, K.: Efficiently solving very large-scale routing problems. Computers & Operations Research **107**, 32 – 42 (2019). https://doi.org/10.1016/j.cor.2019.03.006, http://www.sciencedirect.com/science/article/pii/S0305054819300668
8. Clarke, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Research **12**(4), 568–581 (1964). https://doi.org/10.1287/opre.12.4.568
9. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Management Science **6**(1), 80—-91 (1959)
10. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983). https://doi.org/10.1126/science.220.4598.671
11. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated Local Search, pp. 320–353. Springer US, Boston, MA (2003)
12. Máximo, V.R., Cordeau, J.F., Nascimento, M.C.V.: AILS-II: An adaptive iterated local search heuristic for the large-scale capacitated vehicle routing problem. INFORMS Journal on Computing **36**(4), 974–986 (2024). https://doi.org/10.1287/ijoc.2023.0106
13. Toth, P., Vigo, D. (eds.): The Vehicle Routing Problem. Society for Industrial and Applied Mathematics, Philadelphia, PA (2002)
14. Toth, P., Vigo, D. (eds.): Vehicle Routing: Problems, Methods, and Applications. Society for Industrial and Applied Mathematics, Philadelphia, PA (2014). https://doi.org/10.1137/1.9781611973594, https://epubs.siam.org/doi/abs/10.1137/1.9781611973594