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.

A Hybrid Metaheuristic for the Family Capacitated Vehicle Routing Problem

Bruno Oliveira

bruno_mattos@id.uff.br

Universidade Federal Fluminense

Diogo Lima

Universidade Federal Fluminense

Marcos Roboredo

Universidade Federal Fluminense

Research Article

Keywords: Iterated local search, Vehicle routing problem, Hybrid algorithm, Family capacitated vehicle routing problem, Set partitioning

Posted Date: July 2nd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10084389/v1

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

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Journal of Heuristics on October 16th, 2026. See the published version at https://doi.org/10.1007/s10732-026-09607-3.

A Hybrid Metaheuristic for the Family Capacitated Vehicle Routing Problem

Bruno Oliveira[1*], Diogo Lima[2†] and Marcos Roboredo[2†]

[1*]Programa de Pós Graduação em Engenharia de Produção, Universidade Federal Fluminense, Rua Passo da Pátria, Niterói, 24210240, Rio de Janeiro, Brazil.
[2]Departmento de Engenharia de Produção, Universidade Federal Fluminense, Rua Passo da Pátria, Niterói, 24210240, Rio de Janeiro, Brazil.

*Corresponding author(s). E-mail(s): bruno_mattos@id.uff.br;
Contributing authors: diogofls@id.uff.br; mcroboredo@id.uff.br;
†These authors contributed equally to this work.

Abstract

The *Family Traveling Salesman Problem* (FTSP) seeks a minimum-cost tour from a depot that visits a prescribed number of members of each customer family. This paper studies the *Family Capacitated Vehicle Routing Problem* (F-CVRP), a generalization of the FTSP in which customers carry demands and are served by a homogeneous fleet of capacitated vehicles, and proposes a hybrid metaheuristic for it. The method, named ILS+SP, couples an Iterated Local Search (ILS), which embeds a Random Variable Neighborhood Descent (RVND) over classical and problem-specific neighborhoods and accepts worsening moves through a Simulated-Annealing criterion, with a Set Partitioning (SP) post-optimization over the pool of routes collected during the search. On 144 large benchmark instances with more than 50 customers, the metaheuristic wins or ties on every

instance against the state-of-the-art heuristic of the literature, reducing the average gap to the best known solution from 1.84% to 0.01%. On a new benchmark of 120 larger instances with up to 142 customers, it attains an average gap of 0.47% within an average running time of 37s, demonstrating its scalability.

Keywords: Iterated local search, Vehicle routing problem, Hybrid algorithm, Family capacitated vehicle routing problem, Set partitioning

1 Introduction

The *Family Traveling Salesman Problem* (FTSP) generalizes the Traveling Salesman Problem by grouping the cities into families and requiring only a prescribed number of members of each family to be visited by a single minimum-cost tour from the depot. Since its introduction by Morán-Mirabal et al. (2014), the FTSP has been studied mostly through (meta)heuristics (Wu et al., 2021; Bernardino and Paias, 2021; Nourmohammadzadeh et al., 2023; Chaves et al., 2024; Pandiri and Singh, 2024), reflecting the combinatorial difficulty of choosing both which family members to visit and how to route them.

The *Family Capacitated Vehicle Routing Problem* (F-CVRP), introduced by Bernardino and Paias (2024), lifts the FTSP to a multi-vehicle, capacitated setting: a homogeneous fleet of capacitated vehicles must serve, by depot-rooted routes, the required number of customers of each family while respecting the vehicle capacity. As the instances grow, exact methods become time-consuming, so fast heuristics able to deliver high-quality solutions within tight time budgets are valuable, both as standalone solvers and as primal-bound providers inside exact frameworks.

This paper proposes a hybrid metaheuristic for the F-CVRP, named ILS+SP. Its contributions are as follows:

- We develop an Iterated Local Search (ILS) that embeds a Random Variable Neighborhood Descent (RVND) over a portfolio of classical and problem-specific neighborhoods, and that controls the acceptance of worsening solutions through a

Simulated-Annealing (SA) criterion with a time-driven cooling schedule. A dedicated operator handles the F-CVRP-specific decision of which family members to visit.

- We hybridize the search with a Set Partitioning (SP) post-optimization that re-optimizes over the pool of routes collected during the run, yielding a solution no worse than the best one found by the iterative phase.
- We evaluate the proposed method on the 144 benchmark instances containing more than 50 customers. The results show that ILS+SP matches or outperforms the current state-of-the-art heuristic from the literature (Bernardino and Paias, 2024) on all instances, reducing the average gap to the Best Known Solution (BKS) from 1.84% to 0.01%. Furthermore, on a new benchmark comprising 120 larger instances with up to 142 customers derived from the CVRP set X (Uchoa et al., 2017), ILS+SP achieves an average gap of 0.47% within an average running time of 37 s.

The remainder of the paper is organized as follows. Section 2 reviews related work, with emphasis on heuristic methods. Section 3 formally defines the F-CVRP. Section 4 presents the ILS+SP metaheuristic. Section 5 reports the computational results, and Section 6 summarizes our conclusions.

2 Review of the literature

2.1 Heuristics for the Family Traveling Salesman Problem

The FTSP was introduced by Morán-Mirabal et al. (2014) together with two randomized heuristics: a biased random-key genetic algorithm (BRKGA) and a GRASP with evolutionary path-relinking. A diverse stream of heuristic methods followed. Wu et al. (2021) proposed a DNA-computing algorithm based on the Adleman–Lipton model, encoding solutions on DNA strands to exploit massive parallelism. Bernardino and Paias (2021) developed a population-based metaheuristic and a local-search method, alongside a branch-and-cut hybrid, improving the best-known upper bounds across the benchmark and introducing the instance generator later used to build the F-CVRP benchmark. Nourmohammadzadeh et al. (2023) combined a MILP model

with genetic algorithms and simulated annealing within a POMUSIC matheuristic. Chaves et al. (2024) proposed a parallel branch-and-cut with local search together with a BRKGA whose parameters are tuned online by Q-learning. Finally, Pandiri and Singh (2024) introduced a hyper-heuristic that orchestrates three large-neighborhood-search operators followed by a local-search intensification, matching or improving the best results of Bernardino and Paias (2021) and Chaves et al. (2024).

2.2 Multi-vehicle family routing and solution methods

Four studies extend the FTSP to multiple routes: Bernardino et al. (2022) proposed multi-depot variants in which each depot is served by a single uncapacitated salesman, in soft- and hard-clustered versions, solved by branch-and-cut on instances of up to 200 nodes; Domínguez-Casasola et al. (2024) studied a heterogeneous-fleet variant and proposed a BRKGA with four decoder strategies alongside MILP models; and Cunha et al. (2024) solved a variant with uncapacitated vehicles and bounded per-route visits by an Adaptive Large Neighborhood Search (ALNS), which lies outside the F-CVRP scope as it ignores capacities. The F-CVRP itself was introduced by Bernardino and Paias (2024), who, besides MILP formulations and a branch-and-cut algorithm, devised an ILS heuristic to obtain feasible solutions for the instances left open by their exact methods.

2.3 Positioning of the proposed approach

Among the multi-route extensions of the FTSP surveyed above, the F-CVRP of Bernardino and Paias (2024) stands out as the most natural generalisation: it retains the single depot and the homogeneous fleet that characterise the classical CVRP, simply replacing the mandatory full coverage of every customer by the family-based selective visit requirement. This work therefore focuses on the F-CVRP.

To the best of our knowledge, the only heuristic proposed so far for the F-CVRP is the ILS of Bernardino and Paias (2024), devised to provide feasible solutions for the instances left open by their exact methods. Our ILS+SP adopts the same high-level ILS skeleton and relies on components that are well established for vehicle routing,

ILS itself (Subramanian et al., 2013; Penna et al., 2019; Osorio-Mora et al., 2023), a Simulated-Annealing acceptance criterion in the spirit of the FILO algorithm of Accorsi and Vigo (2021), a time-budget-driven cooling schedule inspired by Sacramento et al. (2019), and a set-partitioning re-optimization over the generated route pool.

It differs from Bernardino and Paias (2024), however, in four main respects. First, it adds a SP post-optimization over the pool of routes collected during the search, a matheuristic phase absent from their method. Second, the choice of which family members to visit is handled differently in the local search: whereas Bernardino and Paias (2024) only swap a visited customer with a non-visited member of the same family *in place*, the incoming customer inheriting the position of the removed one, our *MemberRelocate* neighborhood inserts the incoming non-visited member at its best position over all routes, evaluated by best improvement. Third, the neighborhoods are explored through a RVND that randomizes their search order, rather than the fixed order adopted in Bernardino and Paias (2024). Fourth, worsening solutions are accepted by a SA criterion, instead of the acceptance restricted to improving solutions used there.

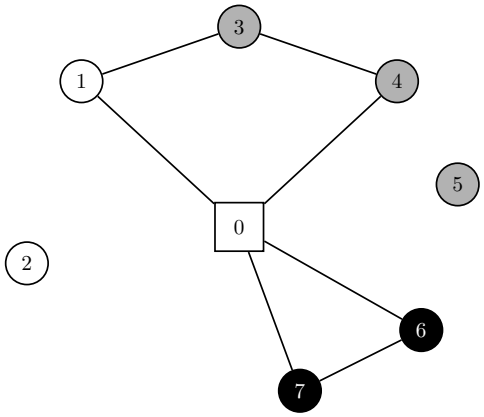
3 The problem

In the F-CVRP, a fleet of K identical vehicles of capacity Q, based at a single depot, must serve a set N of customers partitioned into families. Let $\mathcal{L} = \{1, \ldots, L\}$ index the families, let $F_l \subseteq N$ be the customers of family l (of size $n_l = |F_l|$, with $l(i)$ the family of customer i), and let every customer of family l share a common demand d_l. Each family l specifies a number of required visits $v_l \in \{1, \ldots, n_l\}$. Routing takes place on a complete undirected graph $G = (\{0\} \cup N, E)$ with depot 0 and edge costs c_e for $e \in E$.

A solution is described by a set of routes, each a depot-rooted cycle visiting one or more customers. It is feasible when it uses exactly K routes, serves each visited customer at most once and entirely by a single vehicle, fulfils every family's visit requirement (exactly v_l members of family l are served), and respects the capacity Q

on every route. The cost of a solution is the total cost of the edges of its routes, and the F-CVRP calls for a feasible solution of minimum cost.

Figure 1 shows a feasible solution of a small instance with three families, $F_1 = \{1, 2\}$, $F_2 = \{3, 4, 5\}$, and $F_3 = \{6, 7\}$, demands $d_1 = 4$, $d_2 = 3$, and $d_3 = 2$, visit requirements $v_1 = 1$ and $v_2 = v_3 = 2$, and $K = 2$ vehicles of capacity $Q = 12$. The two routes 0–1–3–4–0 and 0–6–7–0 visit one member of family 1 and both required members of families 2 and 3, so customers 2 and 5 are left unvisited.



l	F_l	n_l	v_l	d_l
1	$\{1, 2\}$	2	1	4
2	$\{3, 4, 5\}$	3	2	3
3	$\{6, 7\}$	2	2	2

Fig. 1: A feasible solution for a small F-CVRP instance with three families. Node shades indicate the families; customers 2 and 5, which lie on no route, are left unvisited.

4 The proposed Hybrid Metaheuristic for the F-CVRP

This section presents the metaheuristic approach proposed in this work, referred to as ILS+SP. The method is structured around an ILS framework, integrating a RVND in the local-search phase and a SP model for solution refinement.

The choice of ILS is justified by its successful application to several VRP variants, as reported in Subramanian et al. (2013), Penna et al. (2019), and Osorio-Mora et al.

(2023). The idea of using a MIP model to refine the pool of routes produced by the heuristic has also been adopted by Subramanian et al. (2013). Differently from these works, we couple ILS SA acceptance criterion, in the spirit of the FILO algorithm of Accorsi and Vigo (2021) for the Capacitated VRP, while the time-budget-driven cooling schedule was inspired by the SA implementation of Sacramento et al. (2019). Whereas perturbations diversify the search, the SA criterion controls the trade-off between intensification and the controlled acceptance of worsening moves, allowing the algorithm to escape from basins of attraction that pure descent or strict acceptance would not abandon.

Let s be an F-CVRP candidate solution. All neighborhoods and perturbations described in this section are designed to preserve, by construction, the exact number of visits per family (v_l for each $l \in \mathcal{L}$) and the requirement that exactly K routes are used. Only the vehicle-capacity constraint may be temporarily violated, and is handled through a penalty term in the objective function. Specifically, let $\mathcal{R}(s)$ denote the set of routes in s and let $q(r)$ denote the total demand served by route r; the objective function used by the heuristic is

$$f(s) = \sum_{r \in \mathcal{R}(s)} c_r + \theta \sum_{r \in \mathcal{R}(s)} \max\{0, q(r) - Q\}, \tag{1}$$

where c_r is the cost of route r, Q is the vehicle capacity, and θ is a penalty coefficient for capacity violations. To promote feasibility throughout the search, θ is dynamically adjusted: after each neighborhood exploration it is decreased by a multiplicative factor $\phi < 1$ when the current solution is capacity-feasible, and increased by the same factor otherwise. By convention, $f(\text{NULL}) = +\infty$.

The overall structure of the proposed ILS+SP algorithm is outlined in Algorithm 1. Its design is inspired by the framework introduced by Oliveira et al. (2025) for VRP variants with lockers.

The algorithm receives three parameters: the initial temperature T_0, the cooling exponent α and the total time budget maxTime. Additional internal parameters (the penalty initial value and multiplier θ, ϕ, the temperature lower bound ε, and the perturbation intensities $p_{\text{swap}}, p_{\text{shift}}$) are reported in Section 5.

During its execution, the algorithm maintains a pool of routes $\mathcal{P}$, initially empty (line 2), and tracks two solutions: the SA reference solution s and the best feasible solution s^* found so far. An initial solution s_0 is generated once by the InitialSolution procedure (line 4) and improved via local search (line 5). All feasible routes of the improved solution are added to $\mathcal{P}$ (line 6), and s is updated whenever $f(s_0) < f(s)$. At the beginning of each iteration, the temperature is computed via a time-dependent cooling schedule (line 14):

$$T \leftarrow \max\Big(T_0 \cdot \Big(1 - \frac{\tau}{\text{maxTime}}\Big)^{\alpha},\ \varepsilon\Big), \tag{2}$$

where τ denotes the elapsed time since the start of the algorithm and $\varepsilon > 0$ prevents the temperature from reaching zero. This schedule was preferred over an iteration-dependent counterpart because the number of iterations is highly heterogeneous across instances: large instances perform fewer iterations per second than small ones, so an iteration-dependent schedule would cool down at very different effective rates depending on instance size, defeating the purpose of using a single value of α.

A perturbed solution s' is then generated from the current reference solution s (line 15) and improved via local search. Its feasible routes are added to $\mathcal{P}$ (line 17), and s^* is updated whenever s' is both feasible and better than the current best (line 18). The SA criterion then decides whether s' becomes the new reference: if accepted, $s \leftarrow s'$; otherwise, s remains unchanged so that the next iteration perturbs s again (lines 21–24). Since SA may accept worse solutions, s can temporarily worsen, allowing broader exploration; s^*, however, is monotonically non-increasing and always records the best feasible solution seen.

Once the time budget is exhausted, the algorithm solves a SP formulation over the pool $\mathcal{P}$ to produce the final solution (line 26). Whenever at least one feasible solution has been found during the search ($s^* \neq$ null), the pool $\mathcal{P}$ contains all the capacity-feasible routes that compose s^*; since no time limit is imposed on the SP, its optimum is by construction at least as good as s^*, so the assignment $s^* \leftarrow$ SetPartitioning($\mathcal{P}$) can be carried out unconditionally. In the rare event that the iterative phase produces

no feasible solution at all, the SP may still recover one from the route pool whenever a feasible combination of routes exists, although this is not guaranteed in general.

Algorithm 1 Main steps of the proposed ILS+SP algorithm

procedure ILS+SP($T_0, \alpha, \text{maxTime}$)
  $\mathcal{P} \leftarrow \emptyset$
  $s \leftarrow$ NULL; $s^* \leftarrow$ NULL ▷ $f(\text{NULL}) := +\infty$
  $s_0 \leftarrow$ InitialSolution()
  $s_0 \leftarrow \text{RVND}(s_0)$
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{\, r \in \mathcal{R}(s_0) : q(r) \le Q \,\}$ ▷ capacity-feasible routes only
  if $f(s_0) < f(s)$ **then**
    $s \leftarrow s_0$
  end if
  if s_0 is feasible **and** $f(s_0) < f(s^*)$ **then**
    $s^* \leftarrow s_0$
  end if
  while $\tau <$ maxTime **do**
    $T \leftarrow \max\left(T_0 \cdot \left(1 - \dfrac{\tau}{\text{maxTime}}\right)^{\alpha},\ \varepsilon\right)$
    $s' \leftarrow$ Perturbation(s)
    $s' \leftarrow \text{RVND}(s')$
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{\, r \in \mathcal{R}(s') : q(r) \le Q \,\}$
    if s' is feasible **and** $f(s') < f(s^*)$ **then**
      $s^* \leftarrow s'$
    end if
    $\Delta \leftarrow f(s') - f(s)$ ▷ stored penalized cost, Eq. (1)
    if $\Delta \le 0$ **or** $\mathcal{U}(0,1) < \exp(-\Delta/T)$ **then**
      $s \leftarrow s'$
    end if
  end while
  $s^* \leftarrow$ SetPartitioning($\mathcal{P}$)
  return s^*
end procedure

The procedure for generating initial solutions is described in Subsection 4.1. The local-search algorithm and its neighborhood operators are presented in Subsection 4.2,

while the perturbation mechanisms are detailed in Subsection 4.4. Finally, the set-partitioning formulation used in the last step is provided in Subsection 4.5.

4.1 Initial solution

The initial solution is built by a Best Parallel Insertion (BPI) heuristic. The procedure starts with exactly K empty routes, each consisting only of the depot, and iteratively inserts unvisited customers into the solution. At each step, the algorithm evaluates all possible insertion positions of all remaining candidate customers across the K routes; the customer-position pair that leads to the minimal increase in $f(\cdot)$ (i.e., taking both route cost and capacity penalty into account) is selected. The process is repeated until exactly v_l customers have been inserted from each family $l \in \mathcal{L}$.

To adapt the BPI to the F-CVRP, once a family $l \in \mathcal{L}$ has reached its number of visits v_l, the customers associated with this family are no longer considered for insertion in any route. The procedure therefore produces a solution that is family-visit feasible and contains exactly K routes by construction; capacity may, however, be temporarily violated and is later addressed by the search.

4.2 Local search and neighborhoods

The local-search procedure incorporates nine neighborhoods, organized into three categories: inter-route (five neighborhoods), intra-route (three neighborhoods), and specialized (one neighborhood).

All neighborhoods preserve the structural feasibility imposed by the family-visit and fleet-size requirements: visits are not added or removed (so the family count v_l is unchanged), and no move that would empty an entire route is allowed (so the number of active routes remains exactly K throughout the search).

4.2.1 Inter-route neighborhoods

The inter-route category comprises five classical neighborhoods:

- N_1 - Swap$(1, 1)$: exchanges a customer from one route with a customer from another route.

- N_2 - Swap$(2, 1)$: exchanges two consecutive customers from one route with a single customer from another route.
- N_3 - Swap$(2, 2)$: exchanges two consecutive customers from one route with two consecutive customers from another route.
- N_4 - Shift$(1, 0)$: relocates a single customer from one route to another.
- N_5 - Shift$(2, 0)$: relocates two consecutive customers from one route to another.

Finding the best move in each inter-route neighborhood requires $\mathcal{O}(|N|^2)$ time, whereas applying the chosen move can be done in $\mathcal{O}(1)$ once the supporting data structures (route representations and incremental cost evaluators) are kept up to date.

4.2.2 Intra-route neighborhoods

The intra-route neighborhoods are defined as follows:

- N_6 - Exchange: swaps the positions of two customers within the same route.
- N_7 - Reinsertion: removes a customer and reinserts it at a different position within the same route.
- N_8 - Or-opt2: relocates two consecutive customers to a different position within the same route.

As for the inter-route operators, finding the best move in each intra-route neighborhood requires $\mathcal{O}(|N|^2)$ time, while applying the chosen move is performed in $\mathcal{O}(1)$.

4.2.3 Specialized neighborhoods

The previously described neighborhoods are standard in the VRP literature and assume that the set of visited customers is fixed. To explicitly account for the possibility of replacing visited customers with non-visited members of the same family, we introduce a specialized neighborhood.

- N_9 - MemberRelocate: removes a visited customer c_1 from its current route and inserts a non-visited customer c_2 from the same family at any position in any route.

Finding the best move requires $\mathcal{O}(|N|^3)$ time for N_9, since it considers all combinations of customer to remove, non-visited replacement, and insertion position. In line with the structural invariant stated above, moves that would leave c_1's route empty are discarded: if c_1 is the only customer of its route, N_9 only reinserts c_2 into that same route, so that exactly K non-empty routes are always maintained.

Because each application of N_9 may simultaneously change both the set of visited customers and the routes' structure, this operator is iterated internally: N_9 is applied repeatedly until no further improvement is found, before control is returned to the outer RVND. In the RVND, this nested behavior is treated as a single neighborhood call, after which N_9 is removed from the active list if (and only if) the call ends without any improvement, as for any other neighborhood.

4.3 The RVND procedure

All neighborhood structures are explored using a best-improvement strategy within a RVND framework. At each outer iteration, the list of neighborhoods is randomly shuffled and explored in that order. If an improving move is found in any neighborhood, the current solution is updated and the full list is restored with a new random order. If no neighborhood yields an improvement, the procedure terminates.

After exploring each neighborhood, the penalty coefficient θ is updated: it is multiplied by ϕ when the current solution is capacity-feasible, and divided by ϕ otherwise. This adaptive mechanism drives the search toward feasibility while still allowing temporary violations.

The pseudocode of the RVND procedure is presented in Algorithm 2.

4.4 Perturbations

The purpose of the perturbation phase is to diversify the search by escaping local minima. At each iteration of the ILS, one of three perturbation mechanisms is selected at random and applied to the current reference solution s:

- P1 - RandomSwap$(1, 1)$: with probability 0.20, applies a Swap$(1, 1)$ move between two randomly chosen routes. This operator is applied p_{swap} times.

Algorithm 2 Main steps of the proposed RVND procedure

```
1:  procedure RVND(solution)
2:      s ← solution
3:      NG ← {N_1, N_2, ..., N_9}                          ▷ List of neighborhoods
4:      while |NG| > 0 do
5:          SHUFFLE(NG)
6:          improved ← false
7:          for ng ∈ NG do
8:              s' ← ng(s)                  ▷ Apply neighborhood with best-improvement
9:              UPDATEPENALTY(θ, φ, s)
10:             if f(s') < f(s) then
11:                 s ← s'
12:                 NG ← {N_1, N_2, ..., N_9};  improved ← true
13:                 break
14:             end if
15:         end for
16:         if ¬improved then
17:             NG ← ∅
18:         end if
19:     end while
20:     return s
21: end procedure
```

- P2 - RandomShift$(1, 0)$: with probability 0.30, transfers a randomly selected customer from one route to a random position in another route. This operator is applied p_{shift} times.
- P3 - RandomMemberRelocate: with probability 0.50, removes a visited customer from a route and inserts a randomly chosen non-visited member of the same family at any position in any route. This operator is applied twice.

The selection probabilities (0.20, 0.30, 0.50) were set by preliminary experimentation and held fixed; the operator intensities p_{swap} and p_{shift} were determined by the calibration procedure described in Section 5.

4.5 Set partitioning model

After the execution of the ILS, a large and diverse pool of routes $\mathcal{P}$ is available. Solving a SP formulation over this pool enables a final global optimization step. Whereas the iterative search improves s through local search and perturbation and tracks the best feasible s^* encountered, it does not ensure that the best combination of routes is selected from the entire pool. By solving an SPP over the collected routes, the algorithm exploits all the information generated by the search.

Let c_r denote the cost of route r, let h_{lr} be the number of customers of family l visited by r, and let $h'_{ir} \in \{0, 1\}$ indicate whether r visits customer i; the binary variable λ_r selects route r. The SP formulation follows:

$$\text{Minimize} \quad \sum_{r \in \mathcal{P}} c_r \, \lambda_r \tag{3a}$$

$$\text{subject to} \quad \sum_{r \in \mathcal{P}} h_{lr} \, \lambda_r = v_l, \qquad \forall l \in \mathcal{L}, \tag{3b}$$

$$\sum_{r \in \mathcal{P}} h'_{ir} \, \lambda_r \leq 1, \qquad \forall i \in N, \tag{3c}$$

$$\sum_{r \in \mathcal{P}} \lambda_r = K, \tag{3d}$$

$$\lambda_r \in \{0, 1\}, \qquad \forall r \in \mathcal{P}. \tag{3e}$$

The objective minimizes the total cost of the selected routes. Constraints (3b) enforce the required number of visits of each family, while constraints (3c) ensure that no customer is visited by more than one selected route. Constraint (3d) enforces the use of exactly K routes, and (3e) defines the domain. As $\mathcal{P}$ contains only capacity-feasible routes, no capacity constraint needs to appear explicitly. Any optimal solution of (3) is feasible for the F-CVRP, and the cost of this solution is, by construction, no worse than that of s^*, since s^* corresponds itself to a feasible combination of routes in $\mathcal{P}$.

5 Computational Results

This section reports the computational experiments carried out to assess the proposed ILS+SP metaheuristic for the F-CVRP. All experiments were run on a single thread of a machine with an Intel Core i7-10700 processor at 2.90 GHz and 16 GB of RAM, under Ubuntu 18.04.2 LTS. The metaheuristic was implemented in the Julia programming language, and CPLEX 20.1 was used to solve the SP model of the post-optimization phase.

The internal parameters of the metaheuristic are summarized in Table 1. The combination $T_0 = 200$, $\alpha = 2$, $p_{\text{shift}} = 3$, and $p_{\text{swap}} = 2$ was obtained through the calibration procedure described in Subsection 5.2.1; the remaining parameters were fixed by preliminary experimentation, to which the algorithm proved largely insensitive. The temperature lower bound $\varepsilon = 10^{-10}$ is a non-tuned numerical safeguard against division by zero in the SA acceptance probability as $\tau \to \text{maxTime}$.

Table 1: Parameters of the ILS+SP metaheuristic.

Symbol	Description	Value
Calibrated parameters (see Subsection 5.2.1)		
T_0	Initial SA temperature	200
α	Cooling exponent	2
p_{shift}	Number of sequential applications of *RandomShift* (P2)	3
p_{swap}	Number of sequential applications of *RandomSwap* (P1)	2
Parameters set by preliminary experimentation		
θ	Initial capacity-violation penalty	100
ϕ	Penalty adjustment factor	0.99
p_{relocate}	Number of sequential applications of *RandomMemberRelocate* (P3)	2
p_{P_1}	Selection probability of perturbation P1	0.20
p_{P_2}	Selection probability of perturbation P2	0.30
p_{P_3}	Selection probability of perturbation P3	0.50
Non-tuned numerical constants		
ε	Temperature lower bound (numerical safeguard)	10^{-10}

5.1 Instance sets

The metaheuristic is evaluated on the benchmark of Bernardino and Paias (2024), the same set used by the state-of-the-art method against which we compare. It derives

from the classical set P of CVRP instances of Augerat et al. (1995), 24 instances with 16 to 101 nodes (depot included), each adapted by the FTSP generator of Bernardino and Paias (2021) to add family structure and visit requirements.

Three family-size configurations are considered for each base instance, with the number of families set to 10%, 30%, and 50% of the customers (rounded to the nearest integer). For each (base instance, configuration) pair, four instances are generated according to different visit patterns. In the *reference* pattern, the number of required visits for each family is drawn uniformly between 1 and the family size. In the *low* pattern, it is drawn uniformly between 1 and the corresponding reference value, whereas in the *high* pattern it is drawn uniformly between the reference value and the family size. Finally, in the *random* pattern, the number of visits is drawn uniformly between the low and high values. This process yields $24 \times 3 \times 4 = 288$ instances, which are publicly available at http://familytsp.rd.ciencias.ulisboa.pt/.

The instances are partitioned by size into 144 *small* instances (up to 50 nodes) and 144 *medium* instances (more than 50 nodes). The latter constitute the benchmark set used for comparison with the literature.

To assess scalability on larger instances, we additionally consider a benchmark derived from the well-known set X of CVRP instances (Uchoa et al., 2017). This benchmark, kindly generated and provided by the authors of Bernardino and Paias (2024), consists of 10 base instances with 102 to 144 nodes (including the depot), adapted to the F-CVRP through the same procedure described above, resulting in 120 additional instances, which are publicly available at https://projects-rbernardino.iseg.ulisboa.pt/.

5.2 Results of the proposed ILS

5.2.1 Parameter calibration

The parameters of the SA acceptance criterion and of the perturbation operators were tuned through a preliminary grid search. The initial temperature T_0 was evaluated over $\{50, 100, 200\}$, the cooling exponent α over $\{1, 2, 3\}$, and the number

of sequential applications of *RandomShift* (p_{shift}) and *RandomSwap* (p_{swap}) during perturbation were each tested over $\{1, 2, 3, 4\}$, yielding a full-factorial design of $3 \times 3 \times 4 \times 4 = 144$ parameter combinations. The grid search was conducted on 21 representative instances—12 from set P (8 from group P60 and 4 from group P101) and 9 from set X (3 from each of groups X101, X129, and X143)—using 3 seeds and a time limit of 5 seconds per run. The combination $T_0 = 200$, $\alpha = 2$, $p_{\text{shift}} = 3$, and $p_{\text{swap}} = 2$ achieved the best overall performance across both instance sets and was adopted for all subsequent experiments. Note that the cooling schedule (2) depends on the elapsed-time fraction $\tau/\text{maxTime}$ rather than on absolute time, so the qualitative behavior of the schedule is invariant under scaling of maxTime. This justifies using the parameters calibrated at a 5-second time limit in subsequent experiments with different time budgets.

5.2.2 Comparison with the literature

In this section, we present a comparison between the proposed metaheuristic and the state-of-the-art heuristic of Bernardino and Paias (2024). To compare running times fairly across hardware, every running time reported by Bernardino and Paias (2024) is converted to our reference machine through the ratio of the single-thread PassMark ratings[1] of the two processors. Since our processor is approximately 1.3 times faster on this single-thread rating, each literature running time is divided by 1.3 before any comparison.

The proposed algorithm was evaluated on all 144 large instances (i.e., the instances with more than 50 nodes, comprising groups P51, P55, P60, P65, P70, P76, and P101). For each instance, the time limit maxTime was set to the equivalent of the average running time reported in the literature for the heuristic algorithm of Bernardino and Paias (2024) divided by 1.3. Both our method and the literature results correspond to averages over 5 random seeds. The BKS for each instance is defined as the minimum cost found across both methods and all seeds. The per-instance solution values of the heuristic of Bernardino and Paias (2024), used both to recompute their gaps against

[1] https://www.cpubenchmark.net

this common BKS and to determine the win/tie/loss counts, were obtained from the supplementary material of Bernardino and Paias (2024).

Table 2 reports, for each group, the number of instances (N), the literature average gap to the BKS and the normalised average running time on the reference hardware (Literature), as well as the best gap, average gap, average consumed running time, and win/tie/loss counts (W/T/L) comparing the best solution of each method over 5 seeds, both for the standalone ILS (ILS) and for the ILS with Set Partitioning post-processing (ILS+SP). All gaps are reported with respect to the BKS, defined as the best solution value found across the literature and all computational experiments performed in this work. In the W/T/L columns, a win corresponds to an instance in which the proposed method's best solution strictly improves over the literature's best solution; a tie corresponds to an instance in which both methods reach the BKS.

The results in Table 2 establish two findings. First, the ILS+SP wins or ties on every one of the 144 instances, with no losses against the literature, achieving average gaps of at most 0.03% in every group. Globally, the average gap to the BKS drops from 1.84% in the literature to 0.01%, and on the largest group (P101), where the literature gap reaches 5.38%, the ILS+SP attains an average gap of 0.00%. Second, even the standalone ILS, without the SP step, is already strongly competitive, with average gaps of at most 0.23% across all groups and a global average of 0.08%. This indicates that the iterative search alone accounts for most of the improvement over the literature, while the SP post-processing closes the remaining small residual gaps left by the iterative phase. Note that the W/T/L counts coincide for ILS and ILS+SP: since the standalone ILS already wins or ties on every instance and the SP can only reduce the cost further, the win/tie classification of each instance is necessarily preserved.

5.2.3 General computational results

To evaluate the impact of the computational budget, the proposed algorithm was tested on all 144 large instances under fixed time limits of 5, 10, 15, and 20 seconds. Each instance was solved with 10 random seeds. For each run, we recorded the average gap before Set Partitioning (ILS), the average gap after Set Partitioning (ILS+SP), and the percentage of runs reaching the BKS. The BKS for each set P instance is

		Literature		ILS				ILS+SP			
Group	N	Avg. (%)	$\bar{t}$ (s)	Best (%)	Avg. (%)	$\bar{t}$ (s)	W/T/L	Best (%)	Avg. (%)	$\bar{t}$ (s)	W/T/L
P51	12	0.43	4.69	0.00	0.02	4.69	12/0/0	0.00	0.00	4.83	12/0/0
P55	48	0.57	4.57	0.02	0.05	4.57	34/14/0	0.00	0.01	4.92	34/14/0
P60	24	0.71	5.05	0.00	0.11	5.05	19/5/0	0.00	0.00	5.27	19/5/0
P65	12	2.34	5.63	0.05	0.23	5.63	11/1/0	0.01	0.03	6.23	11/1/0
P70	12	1.42	9.41	0.03	0.15	9.41	12/0/0	0.00	0.00	10.74	12/0/0
P76	24	4.42	12.76	0.00	0.03	12.76	24/0/0	0.00	0.00	16.19	24/0/0
P101	12	5.38	29.40	0.00	0.03	29.40	12/0/0	0.00	0.00	32.05	12/0/0
Global	144	1.84	8.59	0.01	0.08	8.59	124/20/0	0.00	0.01	9.71	124/20/0

Table 2: Comparison with the literature on the 144 large benchmark instances ($n > 50$).

the same as defined in Subsection 5.2.2: the minimum cost found across the literature heuristic and all ILS+SP runs. All gaps are computed with respect to this BKS.

Table 3 summarises the results by instance group and time limit. The Set Partitioning phase substantially improves solution quality across all groups and all time limits. Groups P51, P55, P60, and P70 reach a 100% BKS rate already at the 5-second budget; group P65 reaches it at 20 seconds, and group P76 at 99.2%. The hardest group, P101, exhibits a clear scalability trend: the ILS+SP average gap decreases from 0.10% at 5 seconds to 0.01% at 20 seconds, and the corresponding BKS attainment rate rises from 76.7% to 93.3%.

Globally, the average ILS+SP average gap decreases from 0.02% at 5 seconds to 0.00% at 20 seconds, while the percentage of runs reaching the BKS rises from 95.6% to 99.3%. Moreover, the best ILS+SP gap is at most 0.01% across all groups and time limits—reaching 0.00% everywhere except P76 at the 5-second budget—indicating that every instance reached the BKS in at least one of the 10 seeds for all but one group-budget combination. Taken together with the results of Subsection 5.2.1, this shows that ILS+SP is both effective, closing every literature gap, and robust to the available computational budget.

We now extend this evaluation to the 120 set X instances. The same four time limits (5, 10, 15, and 20 seconds) and 10 random seeds are used. All gaps are computed against the BKS, defined as the minimum objective value found by our ILS+SP across all seeds and time limits.

Table 4 summarises the results. Compared with the set P experiments, the ILS and ILS+SP average gaps are considerably higher, reflecting the greater difficulty of the larger instances. Nevertheless, the SP post-processing consistently improves upon the ILS solution across every group and every time limit. Globally, the average ILS+SP gap decreases from 0.96% at 5 s to 0.47% at 20 s, while the Best SP gap drops from 0.16% to 0.05%, indicating that the algorithm finds near-optimal solutions for every instance with at least one seed.

Figure 2 shows the average BKS gap before and after the SP post-processing for both benchmarks at each time limit: the SP phase consistently and substantially reduces the ILS gap, cutting it by 81–97% on set P and 33–49% on set X.

Figure 3 contrasts the convergence of both benchmarks: set P ILS+SP gaps are already below 0.03% at 5 s, while set X shows a steeper improvement and reaches 0.47% at 20 s.

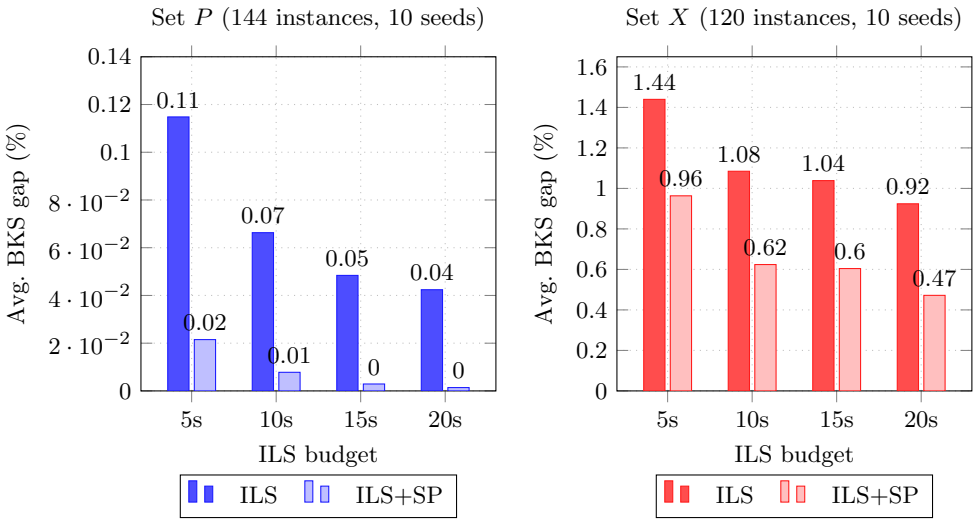


Fig. 2: Average BKS gap before (ILS) and after (ILS+SP) the Set Partitioning post-processing, for set P and set X, averaged over all instances and 10 seeds. A gap of 0.00% indicates that ILS+SP matches the BKS.

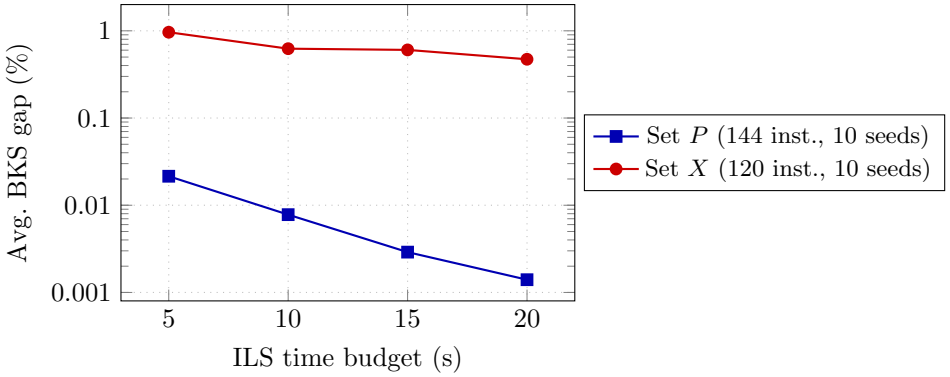


Fig. 3: Average BKS gap of the ILS+SP solution vs. ILS time budget (log scale), for set P and set X, both with 10 seeds. Gaps are averaged over all seeds and instances within each set.

6 Conclusions

This paper proposed ILS+SP, a hybrid metaheuristic for the *Family Capacitated Vehicle Routing Problem* (F-CVRP). The method combines an Iterated Local Search driven by a Simulated-Annealing acceptance criterion with a Random Variable Neighborhood Descent over classical and problem-specific neighborhoods, including an operator that revises which family members are visited, and a Set Partitioning post-optimization over the pool of routes collected during the search.

Across both benchmarks the metaheuristic proved highly effective. On the 144 large set P instances, ILS+SP won or tied on every instance against the literature, cutting the average gap to the best-known solution from 1.84% to 0.01% under the per-instance budget matched to the literature running time; under a uniform 20-second budget, the BKS attainment rate reached 99.3%. On the harder, new larger set X instances, the Set Partitioning phase consistently improved the iterated-search solution across every group and time budget, lowering the global average gap from 0.96% at 5 s to 0.47% at 20 s and the average best-over-seeds gap from 0.16% to 0.05%.

Future work includes extending the metaheuristic to heterogeneous fleets and to variants with time windows or family incompatibilities, designing additional family-aware neighborhoods, and investigating exact solution methods based on branch-cut-and-price algorithm.

Acknowledgments

MR received funding from project CNPq Universal 408562/2025-0. The authors are grateful to the authors of Bernardino and Paias (2024) for kindly providing the set X benchmark instances used in the scalability experiments.

References

Augerat, P., Naddef, D., Belenguer, J., Benavent, E., Corberan, A., Rinaldi, G.: Computational results with a branch and cut code for the capacitated vehicle routing problem (1995)

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

Bernardino, R., Gouveia, L., Paias, A., Santos, D.: The multi-depot family traveling salesman problem and clustered variants: Mathematical formulations and branch-&-cut based methods. Networks **80**(4), 502–571 (2022)

Bernardino, R., Paias, A.: Heuristic approaches for the family traveling salesman problem. International Transactions in Operational Research **28**(1), 262–295 (2021)

Bernardino, R., Paias, A.: The family capacitated vehicle routing problem. European Journal of Operational Research **314**(3), 836–853 (2024)

Cunha, C.B., Massarotto, D.F., Fornazza, S.L., Mendes, A.B.: An alns metaheuristic for the family multiple traveling salesman problem. Computers & Operations Research **169**, 106750 (2024)

Chaves, A.A., Vianna, B.L., Silva, T.T., Schenekemberg, C.M.: A parallel branch-and-cut and an adaptive metaheuristic to solve the family traveling salesman problem. Expert Systems with Applications **238**, 121735 (2024)

Domínguez-Casasola, S., González-Velarde, J.L., Ríos-Solís, Y.Á., Reyes-Vega, K.A.: The capacitated family traveling salesperson problem. International Transactions in Operational Research **31**(4), 2123–2153 (2024)

Morán-Mirabal, L.F., González-Velarde, J.L., Resende, M.G.: Randomized heuristics for the family traveling salesperson problem. International Transactions in Operational Research **21**(1), 41–57 (2014)

Nourmohammadzadeh, A., Sarhani, M., Voß, S.: A matheuristic approach for the family traveling salesman problem. Journal of Heuristics **29**(4), 435–460 (2023)

Osorio-Mora, A., Escobar, J.W., Toth, P.: An iterated local search algorithm for latency vehicle routing problems with multiple depots. Computers & Operations Research **158**, 106293 (2023)

Oliveira, B., Pessoa, A., Roboredo, M.: Hybrid iterated local search algorithm for the vehicle routing problem with lockers: B. oliveira et al. Journal of Heuristics **31**(2), 22 (2025)

Pandiri, V., Singh, A.: Solution of the family traveling salesman problem using a hyper-heuristic approach. Engineering Applications of Artificial Intelligence **133**, 108193 (2024)

Penna, P.H.V., Subramanian, A., Ochi, L.S., Vidal, T., Prins, C.: A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet. Annals of

Operations Research **273**, 5–74 (2019)

Sacramento, D., Pisinger, D., Ropke, S.: An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. Transportation Research Part C: Emerging Technologies **102**, 289–315 (2019)

Subramanian, A., Uchoa, E., Ochi, L.S.: A hybrid algorithm for a class of vehicle routing problems. Computers & Operations Research **40**(10), 2519–2531 (2013)

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., Subramanian, A.: New benchmark instances for the capacitated vehicle routing problem. European Journal of Operational Research **257**(3), 845–858 (2017)

Wu, X., Wang, Z., Wu, T., Bao, X.: Solving the family traveling salesperson problem in the adleman–lipton model based on dna computing. IEEE Transactions on NanoBioscience **21**(1), 75–85 (2021)

		5 s						10 s						15 s						20 s					
Group	N	Avg. gap ILS (%)	Avg. gap ILS+SP (%)	Best gap ILS+SP (%)	BKS_{ILS} (% runs)	BKS_{ILS+SP} (% runs)	$\bar{t}$ (s)	Avg. gap ILS (%)	Avg. gap ILS+SP (%)	Best gap ILS+SP (%)	BKS_{ILS} (% runs)	BKS_{ILS+SP} (% runs)	$\bar{t}$ (s)	Avg. gap ILS (%)	Avg. gap ILS+SP (%)	Best gap ILS+SP (%)	BKS_{ILS} (% runs)	BKS_{ILS+SP} (% runs)	$\bar{t}$ (s)	Avg. gap ILS (%)	Avg. gap ILS+SP (%)	Best gap ILS+SP (%)	BKS_{ILS} (% runs)	BKS_{ILS+SP} (% runs)	$\bar{t}$ (s)
P51	12	0.01	0.00	0.00	95.8	100.0	5.13	0.00	0.00	0.00	100.0	100.0	10.18	0.00	0.00	0.00	100.0	100.0	15.22	0.00	0.00	0.00	100.0	100.0	20.25
P55	48	0.04	0.00	0.00	94.6	100.0	5.28	0.02	0.00	0.00	96.7	100.0	10.44	0.02	0.00	0.00	97.7	100.0	15.59	0.02	0.00	0.00	97.5	100.0	20.71
P60	24	0.06	0.00	0.00	85.8	100.0	5.21	0.05	0.00	0.00	86.2	100.0	10.29	0.02	0.00	0.00	92.1	100.0	15.34	0.02	0.00	0.00	92.5	100.0	20.39
P65	12	0.35	0.01	0.00	65.8	93.3	5.67	0.20	0.01	0.00	70.8	94.2	11.07	0.17	0.00	0.00	72.5	96.7	16.39	0.19	0.00	0.00	72.5	100.0	21.74
P70	12	0.21	0.00	0.00	55.0	100.0	5.80	0.12	0.00	0.00	72.5	100.0	11.19	0.09	0.00	0.00	75.0	100.0	16.50	0.05	0.00	0.00	84.2	100.0	21.73
P76	24	0.15	0.07	0.01	83.3	88.3	5.82	0.07	0.02	0.00	88.8	95.0	11.70	0.05	0.01	0.00	92.5	96.7	18.03	0.02	0.00	0.00	94.2	99.2	24.33
P101	12	0.25	0.10	0.00	49.2	76.7	5.45	0.13	0.04	0.00	63.3	84.2	10.99	0.10	0.01	0.00	66.7	94.2	16.45	0.09	0.01	0.00	68.3	93.3	21.99
Global	144	0.11	0.02	0.00	81.9	95.6	5.44	0.07	0.01	0.00	86.9	97.4	10.77	0.05	0.00	0.00	89.5	98.7	16.14	0.04	0.00	0.00	90.7	99.3	21.50

Table 3: Results by instance group and time limit for the set P benchmark.

		5 s						10 s						15 s						20 s					
Group	N	Avg. gap ILS (%)	Avg. gap ILS+SP (%)	Best gap ILS+SP (%)	BKS_{ILS} (% runs)	BKS_{ILS+SP} (% runs)	$\bar{t}$ (s)	Avg. gap ILS (%)	Avg. gap ILS+SP (%)	Best gap ILS+SP (%)	BKS_{ILS} (% runs)	BKS_{ILS+SP} (% runs)	$\bar{t}$ (s)	Avg. gap ILS (%)	Avg. gap ILS+SP (%)	Best gap ILS+SP (%)	BKS_{ILS} (% runs)	BKS_{ILS+SP} (% runs)	$\bar{t}$ (s)	Avg. gap ILS (%)	Avg. gap ILS+SP (%)	Best gap ILS+SP (%)	BKS_{ILS} (% runs)	BKS_{ILS+SP} (% runs)	$\bar{t}$ (s)
X101	12	0.98	0.22	0.01	16.7	53.3	12.68	0.96	0.21	0.00	16.7	58.3	19.20	0.71	0.06	0.01	22.5	70.8	26.74	0.62	0.06	0.00	27.5	75.0	32.52
X106	12	0.71	0.45	0.08	4.2	21.7	23.84	0.61	0.35	0.09	5.8	28.3	33.25	0.60	0.35	0.05	5.8	28.3	40.96	0.55	0.34	0.14	8.3	35.0	46.36
X110	12	0.72	0.31	0.01	7.5	35.0	10.29	0.57	0.15	0.01	12.5	52.5	18.08	0.44	0.10	0.00	20.0	59.2	24.48	0.45	0.09	0.00	20.8	69.2	31.15
X115	12	2.15	1.95	0.33	37.5	44.2	12.07	1.33	1.12	0.00	45.0	51.7	19.08	1.73	1.54	0.20	52.5	59.2	26.57	1.74	1.31	0.20	53.3	59.2	35.56
X120	12	1.21	1.10	0.12	15.0	17.5	9.20	0.87	0.72	0.05	11.7	21.7	18.98	0.80	0.67	0.01	20.8	30.8	27.07	0.63	0.51	0.01	25.8	37.5	35.22
X125	12	1.41	0.44	0.06	0.8	9.2	22.22	1.18	0.29	0.01	0.8	19.2	30.22	1.12	0.30	0.01	0.8	25.8	37.73	0.92	0.21	0.03	1.7	36.7	44.37
X129	12	2.17	1.12	0.34	0.0	7.5	17.64	1.59	0.76	0.07	4.2	17.5	26.37	1.47	0.65	0.10	4.2	16.7	34.34	1.37	0.41	0.00	3.3	23.3	38.75
X134	12	1.72	1.13	0.23	0.8	1.7	13.18	1.42	0.73	0.11	0.8	10.8	23.53	1.31	0.65	0.10	3.3	20.0	30.36	1.15	0.48	0.05	5.0	24.2	36.22
X139	12	1.48	1.15	0.15	3.3	16.7	10.90	0.97	0.63	0.06	15.0	26.7	21.94	0.85	0.49	0.01	10.8	34.2	29.47	0.90	0.52	0.03	16.7	35.0	38.36
X143	12	1.85	1.75	0.28	15.0	16.7	9.91	1.35	1.27	0.13	19.2	19.2	18.35	1.34	1.22	0.15	22.5	25.8	24.47	0.91	0.80	0.02	22.5	27.5	34.81
Global	120	1.44	0.96	0.16	10.1	22.3	14.19	1.08	0.62	0.05	13.2	30.6	22.90	1.04	0.60	0.06	16.3	37.1	30.22	0.92	0.47	0.05	18.5	42.2	37.33

Table 4: Results by instance group and time limit for the set X benchmark.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- suppILS.pdf