A hybrid matheuristic optimization framework for the Family Capacitated Vehicle Routing Problem

Konstantinos Toulis[a,*], Emmanouil Zachariadis[a]

[a]*Department of Management Science and Technology, Athens University of Economics and Business, 10434, Athens, Greece*

Abstract

This paper introduces a novel solution methodology for the Family Capacitated Vehicle Routing Problem (F-CVRP). The F-CVRP considers customers to be grouped into disjoint clusters called families. Each family requires a predetermined number of customer visits. As a result, the problem jointly calls for the optimal selection of customers to be visited and the generation of the minimum cost set of routes serving all selected customers. To solve the problem, we propose a cyclic matheuristic optimization framework. The main solution refinement mechanism is a hybridization of Tabu-Search and Guided Local Search. At each cycle of the proposed framework the search process originates from new solutions which are produced either by combining node sequences of high-quality solutions, or by solving an MILP subproblem responsible for applying drastic structural modifications by substituting extended customer node sets. The proposed methodology was tested on 384 publicly available F-CVRP benchmark instances. It outperforms previously published exact and heuristic approaches, successfully matching or improving all reported solutions. In specific, it produces better solutions for 47 instances and consistently matches the best solution scores for the rest of 337 instances. It achieves an average solution score improvement of 0.13% with respect to the previously best reported values. The present work also introduces and solves a new collection of 96 large-scale F-CVRP benchmark instances, featuring up to 400 customers, to act as an evaluation tool for future research on the examined problem.

Keywords: vehicle routing, matheuristics, adaptive memory, mixed-integer programming

1. Introduction

This paper explores a variant of the Vehicle Routing Problem (VRP) known as the Family Capacitated Vehicle Routing Problem (F-CVRP), introduced by Bernardino and Paias (2024). In this problem, customers are grouped into predefined clusters, denoted as families, each requiring a fixed number of visits. All customers within the same family share the same product demand. Given a fleet of homogeneous capacitated vehicles, the goal is to construct a set of cost-minimizing routes while

*Corresponding author
Email address: kon.toulis@aueb.gr (Konstantinos Toulis)

ensuring that the required family visit constraints are met. The F-CVRP can be decomposed into two interconnected decision layers: selecting the customer subsets to be served and efficiently sequencing the served customers along the vehicle routes. The two main features of the examined F-CVRP are the clustering of customers and the selective customer service. In the following, we provide relevant research works on vehicle routing problems with the aforementioned characteristics.

The concept of considering nodes as part of families, groups, or clusters has been extensively studied in the vehicle routing literature. Traditional clustering strategies in VRPs can be classified into hard clustering and soft clustering approaches. In hard clustering, all customers within a cluster must be visited consecutively before serving another cluster. Chisman (1975) introduced the Clustered Traveling Salesman Problem (CTSP), where cities are grouped into clusters that must be fully served before proceeding to the next, although the order in which clusters are visited remains open for optimization. Sevaux and Sörensen (2008) generalized the Clustered TSP by introducing the Clustered Vehicle Routing Problem (CluVRP), in which customers belonging to the same cluster must be served sequentially by a single vehicle before it moves to another cluster.

Soft clustering, by contrast, offers increased flexibility by allowing customers from different clusters to be mixed within the same route, although additional constraints may still regulate how service is provided. Defryn and Sörensen (2017) developed a heuristic for the classical, hard-constrained CluVRP and introduced a relaxed variant known as the Soft-CluVRP. This version enables interleaved servicing of clusters, provided that each cluster is entirely handled by a single vehicle. In this context, the F-CVRP examined in this study aligns more closely with the soft clustering variants found in the literature. Subsequent research has extended their work, notably through a branch-and-price approach by Hintsch and Irnich (2020), as well as a Large Neighborhood Search algorithm proposed by Hintsch (2021), which achieved substantial improvements over earlier heuristic techniques.

The customer selection aspect of the F-CVRP bears resemblance to the class of Vehicle Routing Problems with Profits (VRPPs) and their variants, as described by Speranza et al. (2014). For this type of problems, a subset of customers is chosen for service, often with the objective of maximizing profits. Originally, single vehicle profitable routing problems were introduced and examined. The most important such problem is known as the Orienteering Problem (OP) (Tsiligirides (1984)). A particularly relevant generalization of the OP, combining both customer clusters and selective service, is the Clustered Orienteering Problem (COP), presented by Angelelli et al. (2014). Under the COP, clusters are associated with a profit, which may be collected only if all of its customers are served. In their paper, Angelelli et al., implemented both a branch-and-cut and a Tabu-Search algorithm. Computational tests on instances with up to 318 customers indicate significant challenges in finding optimal solutions using exact methodologies. An intriguing variation of the OP that introduces an additional decision layer in customer selection is the Set Orienteering Problem (SOP) (Archetti et al. (2018)). Under the SOP, visiting any node within a cluster is sufficient to collect the entire profit

associated with that cluster. As a result, the SOP requires optimization both in terms of selecting the served customer clusters, and also regarding the selection of the served customer for each covered cluster. In their work, they propose a matheuristic algorithm, utilizing a combination of Tabu-Search and MILP formulations. Computational results on instances with up to 1084 vertices show that the matheuristic produces high-quality results in reasonable computing times. Dontas et al. (2023) combined matheuristics with adaptive memory programming, and managed to match or improve the best known upper bounds in 98.2% of published instances. Further expanding upon the established literature of the SOP, Archetti et al. (2024) later proposed a novel mathematical formulation and branch-and-cut algorithm for the SOP, which performed favorably against previous exact approaches. Note that the SOP model is related to the problem examined in the present paper as per both the selective customer service and the organization of customers in clusters. However, contrary to the F-CVRP examined in the present paper, SOP is a single-vehicle problem.

The Team Orienteering Problem (TOP), introduced by Butt and Cavalier (1994), extends the classical Orienteering Problem (OP) by considering multiple vehicles. Under TOP, a subset of the customers must be selected and assigned to the available vehicles, to maximize the collected profit. Maximum time duration constraints are imposed for each vehicle route implicitly calling for optimizing the produced route set. A TOP extension, referred to as the Capacitated Team Orienteering Problem (CTOP), introduced by Archetti et al. (2009), imposes additional capacity constraints for the vehicles. In their study, Archetti et al. assess several heuristics against branch-and-price methods on datasets containing between 51 and 200 customers. The proposed algorithms consistently produced optimal or near-optimal solutions and proved to be especially effective on instances where exact methods struggled to achieve optimality within a reasonable time frame. The multi-vehicle generalization of SOP has been introduced by Nguyen et al. (2025) and is referred to as the Set Team Orienteering Problem (STOP). Their work proposes a branch-and-price algorithm capable of solving small instances to optimality, whereas a Large Neighborhood Search (LNS) approach is employed for larger instances. Their methods are also tested on SOP instances and outperform previous approaches, regarding the solution quality. The STOP shares many structural similarities with the F-CVRP, particularly in its consideration of multiple vehicles, customer clustering, and selective service requirements.

In the context of the examined F-CVRP, families represent clusters which require a predetermined number of visits and whose customers share identical demand values. Clusters with such characteristics were firstly introduced in the Family Traveling Salesperson Problem (FTSP) (Morán-Mirabal et al. (2014)). FTSP calls for the generation of the optimal route for a salesperson that serves customer families by visiting a predetermined number of customers in each of them.

The F-CVRP generalizes the classical Capacitated Vehicle Routing Problem (CVRP), Christofides (1976), thus it constitutes an NP-hard optimization problem. Given its complexity, exact methods struggle with scalability issues, making heuristics valuable particularly when dealing with large prob-

lem instances. The work of Bernardino and Paias (2024) was the first to examine and solve the F-CVRP. They employ a collection of mixed-integer linear programming formulations within a branch-and-cut framework. They also propose an iterated local search (ILS) approach as a complementary strategy for the largest scale datasets. Their computational study considers instances of up to 100 customers. Even for these medium-sized problems, the results obtained indicate that the exact approaches struggled to deliver high-quality solutions within reasonable time limits. The heuristics employed were partially successful in improving the upper bounds in more challenging problem instances.

In this paper, we introduce an efficient matheuristic optimization framework for the F-CVRP, combining algorithmic components that have demonstrated strong performance in related routing problems. Our approach is built upon an adaptive memory programming framework, inspired by its successful application to the Vehicle Routing Problem with Simultaneous Pick-ups and Deliveries by Zachariadis et al. (2010), as well as the SOP tackled by Dontas et al. (2023). In addition to heuristic components, the proposed algorithm leverages exact methods and mixed-integer programming (MIP) formulations to solve subproblems aiming at diversifying the explored solution space. Similar matheuristic strategies have been effectively applied to the Production Routing Problem (PRP) by Manousakis et al. (2022) as well as the SOP by Dontas et al. (2023). Our method integrates heuristics, adaptive memory programming (AMP), and MIP-based routines into a unified framework. The algorithm achieves competitive results, matching or improving all of the best-known solutions reported in the literature. The main contributions of this work are summarized as follows:

- A hybrid matheuristic optimization framework for the F-CVRP that combines two solution construction strategies - leveraging an Adaptive Memory (AM) structure and Integer Programming (IP) models for solving sub-problems - with a Guided Tabu Search procedure for refining solutions.
- Improved upper bounds for 12.2% (or 47) of published instances where the global optimum has not been calculated.
- A new dataset of large-scale F-CVRP instances with up to 400 customers.
- Analytical solution results that will facilitate future research on the examined problem.

The remainder of this paper is organized as follows: In Section 2, we present a formal problem formulation that integrates family-based constraints into the CVRP framework. Section 3 describes the various algorithmic components of the proposed optimization methodology. The computational results of the algorithm are presented in Section 4, along with comparisons against previously published results. Finally, Section 5 concludes the paper.

2. Problem Description and Formulation

The F-CVRP offers a modeling framework for real-world order picking scenarios, where items are grouped into families based on shared characteristics such as product type. In such contexts, only a subset of items from each family may be required to fulfill orders, while constraints like vehicle capacity and routing efficiency remain critical. In their survey, Boysen et al. (2019) examine the adaptations of warehouse management in the e-commerce market. Warehouses operating in this sector receive large volumes of orders, which are, individually, small in size and their items are not costly in terms of storage. To service the incoming demand, they often keep in storage a vast variety of Stock Keeping Units (SKUs). Since orders need to be assembled and delivered in a tight time schedule, these warehouses tend to adopt a scattered storage policy, spreading products of the same SKU to multiple points inside the depot. As a result, the picker delegated with assembling an order will, on average, spend less time in transit trying to collect the various items. F-CVRP can effectively model this operational scenario; with families representing SKUs and customers being the individual items spread in the warehouse. Given a certain number of pickers (vehicles) with a maximum carrying capacity, the F-CVRP calls for the generation of the optimal picker routes which originate from a central warehouse location, then exhaustively collect the ordered items and finally return to the central warehouse location, so that the orders are assembled.

In accordance with most VRP variants, the F-CVRP considers a set of nodes $N = \{1, 2, ...n\}$ representing a network of customers that need to be served. Node 0 denotes the depot, a central location acting as the origin and terminating point of the generated routes. Customers may be placed to any of the K available routes, where K is the number of identical vehicles at our disposal. Each route is associated with one vehicle, which may carry a maximum capacity of Q.

In addition to traditional VRP model parameters, the F-CVRP considers a set of families $L = \{0, 1, ...L\}$, where each family $l \in L$ consists of a subset of nodes $F_l \subseteq N$. These are disjoint sets, meaning that each customer exclusively belongs to a single family. Additionally, customer demand among members of the same family is uniform, such that $d_i = d_l, \forall i \in F_l, \forall l \in L$. Each family is associated with a predetermined number of required visits, denoted as v_l, such that $v_l \leq |F_l|, \forall l \in L$.

We model the F-CVRP as a graph-based optimization problem. Let $G = (V, A)$ be a complete directed graph, where $V = \{0\} \cup N$ is the complete node set made up of the customers and the depot. The arc set $A = \{(i, j) \mid i, j \in V, i \neq j\}$ connects all pairs of nodes. Each arc $(i, j) \in A$ is associated with the required travel cost denoted as c_{ij}. Additionally, let $W \subseteq V$ be an arbitrary subset of vertices. The in-arcs and out-arcs of W are defined as $\delta^-(W) = \{(i, j) \in A : i \notin W, j \in W\}$ and $\delta^+(W) = \{(i, j) \in A : i \in W, j \notin W\}$. In terms of the decision variables, let x_{ij}^k be equal to 1, iff vehicle k traverses arc $(i, j) \in A$. In addition, y_i is equal to 1, iff customer $i \in N$ is visited.

Lastly, we introduce the continuous variable u_i, with $u_0 = 0$ denoting the depot, which defines the order in which vertex i is visited on a tour, as per the Miller-Tucker-Zemlin (MTZ) formulation

(Miller et al. (1960)).

In the following we provide an IP formulation of the F-CVRP:

$$\min \sum_{k=1}^{K} \sum_{(i,j)\in A} c_{ij} x_{ij}^{k}. \tag{1}$$

Subject to:

$$\sum_{k=1}^{K} \sum_{(i,j)\in\delta^{+}(i)} x_{ij}^{k} = y_i, \quad i \in N \tag{2}$$

$$\sum_{k=1}^{K} \sum_{(j,i)\in\delta^{-}(i)} x_{ij}^{k} = y_i, \quad i \in N \tag{3}$$

$$\sum_{k=1}^{K} \sum_{(0,j)\in\delta^{+}(0)} x_{0j}^{k} = K \tag{4}$$

$$\sum_{k=1}^{K} \sum_{(i,0)\in\delta^{-}(0)} x_{i0}^{k} = K \tag{5}$$

$$\sum_{i,j\in N} d_i x_{ij}^{k} \leq Q, \quad \forall k \in \{1, 2, \ldots, K\} \tag{6}$$

$$\sum_{i\in F_l} y_i = v_l, \quad \forall l \in L \tag{7}$$

$$u_j \geq u_i + 1 - n \cdot (1 - x_{ij}^{k}), \quad \forall i, j \in N, i \neq j, \forall k \in \{1, 2, \ldots, K\} \tag{8}$$

$$x_{ij}^{k} \in \{0, 1\}, \forall (i, j) \in A \tag{9}$$

$$y_i \in \{0, 1\}, \forall i \in N \tag{10}$$

$$1 \leq u_i \leq n, \forall i \in N \tag{11}$$

$$u_0 = 0 \tag{12}$$

The objective function 1 calls for the minimization of the travel costs of the generated routes. Constraints 2, 3 represent the flow conservation, meaning that the number of incoming edges of vertex i should equal the number of outgoing edges . Equations 4, 5 ensure that exactly K vehicles leave and return to the depot. Constraint 6 defines that each vehicle's load will not surpass the maximum capacity Q, while constraint 7 dictates that the sum of all customers routed for each family is equal

to the necessary requirement v_l, $\forall l \in L$. We utilize the MTZ formulation to introduce a subtour elimination constraint in 8. Essentially, if node j is serviced by vehicle k which has previously visited node i then $u_j > u_i$, thus preventing subtours or circles that exclude the depot. Finally, with 10 we ensure that each customer is visited at most once.

After presenting the problem formulation and defining its parameters, we now illustrate a valid solution to an F-CVRP instance. The chosen instance is P-n40-k5_4_3_1 and has the following parameters: The customer set is $N = \{1, 2, .., 39\}$, with family parameters as shown in Table 1. Given $K = 3$ vehicles, with a capacity of $Q = 140$, a valid solution is presented in Figure 1.

l	F_l	v_l	d_l
0	{1,.., 7}	6	17
1	{8,..,17}	3	16
2	{18,..31}	3	16
3	{32,..,39}	8	16

Table 1: Family parameters

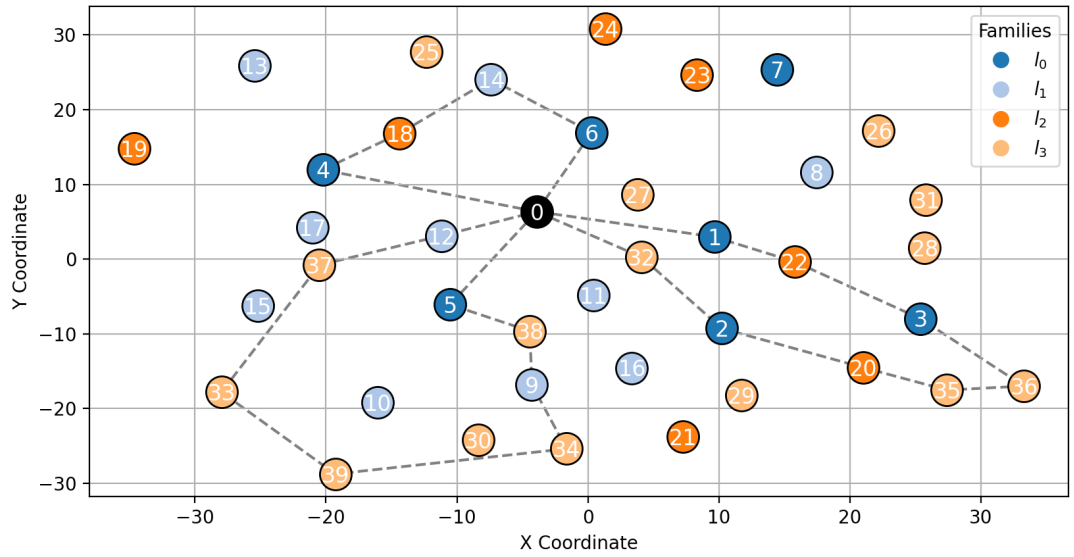

Figure 1: Solution of sample instance

3. Methodology

This section outlines the proposed methodology employed to address the F-CVRP, beginning with a high-level overview of the algorithmic components. We then delve into the initial solution construction heuristic, followed by a description of the Guided Tabu Search (GTS) algorithm. The move operators and the applied diversification techniques are presented in detail. The final part of the present Section introduces two strategies for generating new solutions.

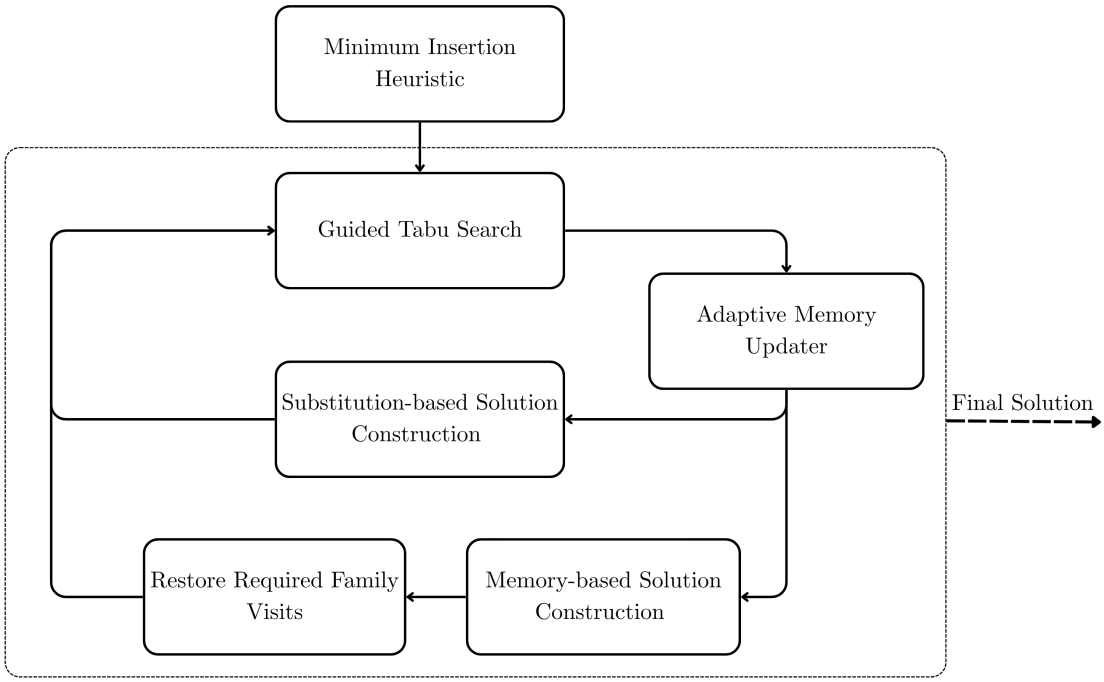

Figure 2: Component Overview

3.1. Component Overview

The proposed optimization methodology is a cyclic framework, with the GTS serving as the basic mechanism for solution refinement. Starting from a solution S, the GTS explores the solution space using four distinct move operators. The incorporation of a tabu policy helps steer the search, allowing for effective intensification within the solution's neighborhood while avoiding cycling and premature convergence. The best solution generated through the GTS method S^*, updates an adaptive memory structure responsible for storing routes of high-quality solutions.

The last step of each cycle provides the starting solution for the next cycle. A new solution is generated through the use of either of two strategies. The first operates by exploiting node sequences which belong to elite solutions observed through the search process structure. These sequences are stored into an Adaptive Memory component and are extracted to form the basis of the next cycle's solution. The second strategy introduces substantial changes to the incumbent solution by making multiple customer substitutions. Both methodologies rely on solving appropriately defined IP subproblems. The selection of the solution construction strategy is randomly decided by variable *memProb*.

The overall algorithmic framework is presented in Figure 2.

Note that the initial solution for the first cycle is built (*ConstructInitialSol()*) by an procedure which starts off by opening K empty vehicle routes. Then, customers are iteratively inserted into the solution according to the minimum objective function change, until a complete solution is formed. If the tightness of the capacity constraints does not allow the generation of a feasible solution, we resort to the best fit method for packing feasibly all required customers into the generated solution. In this

Algorithm 1: Cyclic Optimization Framework Overview

$S \leftarrow ConstructInitialSol()$, $S_{final} \leftarrow \emptyset$, $nonImprove \leftarrow 0$,
while $nonImprove \leq maxRestarts$ **do**
  $S^* \leftarrow GTS(S)$
  if $Z(S^*) < Z(S_{final})$ **then**
    $S_{final} \leftarrow S^*$
    $nonImprove \leftarrow 0$
  end if
  else
    $nonImprove++$
  end if
  $UpdateMemory(S^*)$
  Randomly decide: $memProb \in \{true, false\}$
  if $memProb$ **then**
    $S \leftarrow MemorySol()$
  end if
  else
    $S \leftarrow SubSol(S^*)$
  end if
end while
return S_{final}

case, the insertion position of every customer within the assigned vehicle route is determined so as to minimize the insertion cost.

The overall flow of the proposed methodologies is presented through the Algorithm 1, with $Z(S)$ representing the objective cost of a solution S, computed as the sum of travel costs over all routes in the solution.

3.2. Local Search

Local search serves as the core refinement procedure, operating iteratively on a given solution S with a limit on non-improving moves, and returning the best-found solution S^*. This hybrid approach, called Guided Tabu Search (GTS), combines arc penalizations and a tabu policy with aspiration criteria.

3.2.1. Operators

The GTS method iteratively applies moves to the incumbent solution. At each iteration one of the four local search operators employed is stochastically selected. The neighborhood of the selected

operator is exhaustively explored and the lowest-cost, feasible, and admissible (w.r.t. the tabu policy discussed in Section 3.2.2) move is identified and performed. The four operators are listed in the following:

1. 1-0 exchange: Performs a customer relocation. This relocation can take place within the same route, or between a route pair.
2. 1-1 exchange: Swaps the service positions of a pair of customers contained either in the same route, or in different ones.
3. 2-opt: Substitutes a pair of arcs of the solutions. When the arcs belong to the same route, the intermediate leg is reversed. On the contrary, when the arc pair belongs to two different routes, then a route crossover takes place
4. In-Out exchange: Substitutes a customer present in S with another customer of the same family which obviously is not present in S.

Each operator is assigned a selection probability. For in-out exchanges, the probability is set to $inOutProb = 0.5 * (1 - \frac{|P|}{|N|})$, where P denotes the set of customers in the solution.

By making it proportional to the number of customers that need to be visited out of the total, we effectively regulate the frequency of this move type. Preliminary experiments indicated that this yields a balanced behavior: for instances requiring most of the customers to be included in the solution, the operator is employed scarcely, as there is no strong need to diversify the served customer set. Conversely, for instances that rely more heavily on optimal customer selection, the operator is used more frequently. The rest of the move types share equally the remaining probability.

3.2.2. Tabu Policy

Tabu Search (TS) is a memory-driven optimization technique within the family of local search methods, initially introduced by Glover (1986). The implemented tabu policy restricts the reinsertion of previously removed solution arcs and incorporates an effective aspiration criterion to override prohibitions when beneficial moves are identified. It manages to effectively diversify the search while eliminating the risk of cycling and has proven effective for many VRP variants (Zachariadis et al. (2015), Zachariadis and Kiranoudis (2011)). It works as follows: When a move is performed, it destroys a set of arcs present in the incumbent solution. Let's assume that the arc set A_{rem} is removed from solution S as a result of a move application. Each arc is associated with the $Z(S)$ cost ($p_{(i,j)} \leftarrow Z(S), \forall (i,j) \in A_{rem}$). On subsequent iterations, moves that would introduce an arc set A_{int} to form a modified solution S_{mod} are prohibited, unless $Z(S_{mod}) < p_{(i,j)}, \forall (i,j) \in A_{int}$. To maintain search flexibility and prevent an overly restrictive tabu behavior, the arc cost labels (or *promises*) are periodically reinitialized ($p_{(i,j)} \leftarrow +\infty, \forall a \in A$) every time a fixed number of iterations are completed.

3.2.3. Guided Tabu Search

Algorithm 2: Guided Tabu Search

1 $S^* \leftarrow S$, $i \leftarrow 0$, $nonImprove \leftarrow 0$, $promises \leftarrow \infty$, $penalizedCostMatrix \leftarrow costMatrix$
2 **while** $nonImprove \leq maxNonImprove$ **do**
3 Randomly decide: $g \in \{true, false\}$
4 **if** g **then**
5 $penalizedCostMatrix \leftarrow PenalizeArcs(S)$
6 $move \leftarrow SelectMove(promises, penalizedCostMatrix)$
7 **end if**
8 **else**
9 $move \leftarrow SelectMove(promises, costMatrix)$
10 **end if**
11 $S \leftarrow ApplyMove(S, move)$
12 $promises \leftarrow Update(move)$
13 **if** $Z(S) < Z(S^*)$ **then**
14 $S^* \leftarrow S$
15 $nonImprove \leftarrow 0$
16 **end if**
17 **else**
18 $nonImprove++$
19 **end if**
20 **if** $\bmod(i, promiseIter) = 0$ **then**
21 $promises \leftarrow \infty$
22 **end if**
23 $i++$
24 **end while**
25 return S^*

The proposed Guided Tabu Search framework iteratively applies structural modifications to the candidate solution. To avoid cycling and induce diversification, the tabu policy presented in Section (3.2.2) is used. In addition, the search is controlled by an objective function modification strategy based on the Guided Local Search strategy (Voudouris and Tsang (1996)). The proposed framework is presented in 2. At each iteration, we determine whether the neighborhoods will be explored using the penalized costs with probability g or the actual ones with probability $1 - g$ (line 3). Then, one of the four local search operators is randomly selected and the best feasible cost move which respects the tabu policy is identified. This move is evaluated based on either the penalized cost matrix (line 6) or the original cost matrix (line 9), depending on the earlier decision. The tabu arc costs, referred

to as promises, are periodically reinitialized (line 21).

The cost of an arc (i, j) is penalized (line 5) as per the following expression:

$$c^*_{ij} = (1 + \lambda \cdot p_{ij}) \cdot c_{ij},$$

where p_{ij} represents the number of times arc (i, j) has been penalized and λ is the penalty coefficient. By increasing the cost arcs, the search process is encouraged to eliminate them from the solution. To determine the set of arcs to be penalized, we identify the highest percentage, $topArcs$, of the most costly arcs in the incumbent solution.

3.3. Solution Construction Strategies

The initial solution for every cycle of the proposed optimization framework is generated by two different strategies that were previously briefly introduced in 3.1. The first strategy relies on building a solution from elite node sequences stored in the AM, so it is referred to as Population-based Solution Construction (PSC). The second strategy involves extended removal and insertion operations on a provided solution, so it is named Substitution-based Solution Construction (SSC). The selection of the construction strategy at each cycle is determined probabilistically, controlled by the parameter $memProb$. This hybrid approach allows the algorithm to refine solutions that incorporate previously identified high-quality features, while it allows the exploration of less-examined regions of the solution space through multiple customer substitutions. In the following subsections, the following are discussed:

- the Population-based Solution Construction in which the Adaptive Memory structure and the mechanism behind generating a solution from the node sequences stored in it are presented in detail
- the Substitution-based Solution Construction which relies on the solution of a suitably defined IP model

3.3.1. Population-based Solution Construction

The AM structure is populated by routes found in elite solutions identified through the search process. These routes collectively form a memory pool of fixed maximum size $poolSize$. Routes maintained in the pool are sorted in ascending order based on the objective function of their respective solution. By restricting the pool size, we ensure that only the routes of the minimum cost solutions are preserved in the AM. Preliminary algorithm executions demonstrated that an effective performance was achieved when the $poolSize$ is set to $5 \cdot K$, meaning the AM stores routes from no more than five F-CVRP solutions at any given time. The goal of storing these routes is to identify and extract promising components that can form the basis of new, high-quality solutions in previously unexplored areas of the search space, to be then improved by the GTS procedure.

The routes of the best solution S^* produced at each cycle by the GTS method are compared against the existing AM entries. If the routes are not already in the AM, they are inserted as per their solution objective. If they are present, their associated solution objective is updated to reflect the minimum cost solution ever seen to contain this particular route. Obviously, if new routes are inserted in the pool, the AM size is trimmed to $poolSize$.

The solution routes stored in the Adaptive Memory (AM) consist of variable-length subsequences of visited nodes, which we refer to as *chains*. These chains capture meaningful segments of high-quality routes and serve as reusable building blocks during the reconstruction of new solutions. Thus, from the collection of solution routes, a set of chains C is generated. Every chain $c \in C$ is associated with the following characteristics:

- len_c: the length of the vertex sequence
- $freq_c$: the frequency of the chain in the AM
- sel_c: the number of times it has been selected to be used for building a new solution
- $adjCost_c$: the cost of the chain over the number of nodes contained in it

With the underlying AM structure for PSC established, the strategy proceeds in two main stages: initially, chains are selected from the AM to assemble preliminary vehicle routes; subsequently, an IP model is solved to optimally insert additional customers into these routes, ensuring that all visit requirements are fulfilled.

The Chain Selection methodology is responsible for selecting the appropriate chains that will form the basis of a partial solution S_p. It is an iterated process, repeating a total of K times to generate a set of preliminary vehicle tours. For each iteration, a maximum chain length ($maxLen$) is randomly determined within the range:

$$0.5 \cdot avgLen \leq maxLen \leq 0.7 \cdot avgLen,$$

where $avgLen$ is derived by the average observable route length in previous F-CVRP solution routes stored in the AM. Every chain $c \in C$ is considered eligible for selection if it meets the following criteria:

- $len_c \leq maxLen$
- None of the customers in the chain are already present in S_p
- Including the chain in S_p does not cause any family's required number of visits to be exceeded

The eligible candidate chains are first sorted in descending order of len_c. In the case of ties, candidates are then ranked by $freq_c - sel_c$ in descending order, prioritizing chains that occur frequently

in memory, as this typically indicates high quality. To prevent the repeated use of the same chains, this metric is balanced by a counter tracking how many times each chain has already been selected through this process. If ties still persist, they are resolved by sorting the remaining candidates in ascending order of $adjCost_c$, preferring chains with lower costs relative to their length. The best candidate chain based on this multi-criteria ranking is selected to serve as the foundation for a new route in S_p.

In order to create the partial solution S_p from the selection of chains, the depot is appended to the beginning or end of any chain that does not already include it. Each resulting partial route is then added to S_p. After K such routes have been formed, we employ an IP model called *Restore Required Family Visits (RRFV)* which identifies optimal customer insertions, to satisfy the family constraints of the problem. For every customer $i \in P'$, let $add_{i,k}$ denote a binary decision variable equal to 1, iff node i is to be inserted in route k. Let $c_{i,k}$ represent the minimum cost for inserting customer i in route k. This is pre-calculated by checking all possible insertion positions and identifying the one yielding the minimal cost increase. Let req_l denote the number of members of family $l \in L$ that need to be pushed to the solution, to satisfy the family visit requirements. Additionally, let D_k represent the total load carried by vehicle k, as per solution S_p.

The RRFV formulation calls for:

$$\min \sum_{k=1}^{K} \sum_{i \in P'} c_{i,k} \cdot add_{i,k} \tag{13}$$

subject to:

$$\sum_{k=1}^{K} \sum_{i \in P'} add_{i,k} \leq 1 \tag{14}$$

$$D_k + \sum_{i \in P'} d_i \cdot add_{i,k} \leq Q, \forall k \in \{1, 2, ..., K\} \tag{15}$$

$$\sum_{k=1}^{K} \sum_{i \in F_l \cap P'} add_{i,k} = req_l, \forall l \in L \tag{16}$$

Objective function 13 minimizes the total cost for all customer insertions. Constraints 14 guarantee that each customer is inserted no more than once. Constraints 15 enforce that the vehicle capacity constraints are respected. Lastly, constraints 16 ensure that the selected insertions fulfill the outstanding family visit requirements. It is important to highlight that the minimum insertion costs used in this context are approximations. The estimated cost ($c_{i,k}$) is the actual one for a single customer insertion between a pair of nodes of S_p. However, this is not the case when multiple customer insertions are made into the same insertion position. This deviation is seen not to be inherently disadvantageous: it induces diversification by modifying the solution more drastically.

3.3.2. Substitution-based Solution Construction

The *Substitution-based Solution Construction (SSC)* strategy solves an appropriately defined IP model, named *Multiple Customer Substitution (MCS)*. It is aimed at substituting a subset of customers contained in the best solution S^* identified by the last GTS execution. Let P denote the customers served by S^*. The model's decision variables and parameters are:

- $s_{i,j}$ denotes a binary variable equal to 1, iff node $i \in P$ is substituted by node $j \in P'$.
- r_i denotes a binary variable equal to 1, iff node $i \in P$ is removed from the solution.
- a_j denotes a binary variable equal to 1, iff node $j \in P'$ is added to the solution.
- D_k denotes the total load of vehicle k as per the S^* solution.
- $maxSub$ is a model parameter that sets the upper and lower bounds for the number of substitutions applied to S^*.
- $subc_{i,j}$ represents the penalized substitution cost of i with j, calculated as:

$$subc_{i,j} = addCost(j) - removeCost^*(i), \quad removeCost^*(i) = \frac{removeCost(i)}{1 + \tau \cdot timesRemoved(i)}$$

By adjusting the customer removal costs inversely proportional to the times they have been previously replaced, multiplied by coefficient τ, we discourage the MCS model from repetitively extracting the same customer subsets from the solution. Note that as in the case of 3.3.1, the substitution costs of the MCS model are approximated according to the structure of solution S^*. If substitutions take place in consecutive solution positions, the actual costs changes may may significantly differ from the approximated ones.

In the following, we provide the MCS formulation:

$$min \sum_{i \in P} \sum_{j \in P'} subc_{i,j} \cdot s_{i,j} \tag{17}$$

subject to:

$$D_k + \sum_{i \in P} \sum_{j \in P'} (d_j - d_i) \cdot s_{i,j} \leq Q, \quad \forall k \in \{1, 2, ..., K\} \tag{18}$$

$$\sum_{i \in F_l} r_i = \sum_{j \in F_l} a_j, \forall l \in L \tag{19}$$

$$\sum_{j \in P'} s_{i,j} = r_i, \forall i \in P \tag{20}$$

$$\sum_{i \in P} s_{i,j} = a_j, \forall j \in P' \tag{21}$$

$$0.75 \cdot maxSub \leq \sum_{i \in P} \sum_{j \in P'} s_{i,j} \leq maxSub \tag{22}$$

The objective function 17 of the MCS model aims to minimize the total cost of customer replacements. Constraints 18 ensure that vehicle capacities are respected, whereas constraints 19 dictate that the number of customers of the same family removed and inserted is equal. Constraints 21 and 20 restrict each customer to be added or removed at most once, respectively. Lastly, constraints 22 set a range for the number of performed substitutions. The lower bound plays a crucial role. Without its use, the objective function 17 would drive the model toward making few or no substitutions at all.

4. Computational Results

The computational evaluation of the proposed methodologies was conducted on a system equipped with an Intel Core i7-8700 CPU running at 3.2GHz, 16GB of RAM, and a 64-bit Windows 11 OS. The proposed methodologies were developed in C# using the .NET 9 framework. For solving the IP sub-problems, the Gurobi 12.0.2 Optimizer was employed. All computational experiments were performed in single-threaded mode. To ensure reproducibility and consistency, a fixed seed was used for the random number generator. In the following, we provide a detailed description of the benchmark data sets, an overview of the algorithm's parameters and their default values and present the results obtained from this computational setup on the aforementioned data sets.

4.1. Benchmark Data Sets

The proposed methodologies were tested on a benchmark data set of 384 instances constructed by Bernardino and Paias (2024), available at `https://familytsp.campus.ciencias.ulisboa.pt/cftsp-instances/`. This data set includes small to medium-sized instances, ranging from 15 to 100 customers, with various configurations w.r.t. the number of families, family visit requirements, customer demands, number of vehicles, and other parameters. We note that when examining the reported results of the algorithm of Bernardino and Paias (2024), we noticed a mismatch between the instance file names and the instance identifiers reported in the score tables. This inconsistency has been resolved in our study, to ensure clarity and facilitate future research on the examined problem. Throughout our results and analysis, we adhere strictly to the naming convention used in the instance files, allowing for unambiguous reference and easier cross-comparison.

The original data set consists of symmetric and asymmetric instances. The symmetric instances were derived from the CVRP data set of Augerat (1995). They examine 24 different customer node sets that contain 15 to 100 nodes. Asymmetric instances originate from the data set introduced by

Fischetti et al. (1994). They include eight node sets, with the customer count ranging from 33 to 70 nodes. The FTSP instance generator introduced by Bernardino and Paias (2021) was then applied to the aforementioned instances to create customer families. For each node set, the number of families is set to three distinct values as follows: $|L| \in \{0.1 \cdot |N|,\ 0.3 \cdot |N|,\ 0.5 \cdot |N|\}$, rounded to the nearest integer. Then, for each node set and family number pair, the authors use four distinct patterns to generate the family visit requirements. For each $l \in L$ the number of required visits is generated as follows:

1. reference pattern: v_l is uniformly distributed within $[1, |F_l|]$. Let this number of required visits be denoted as ref_l.
2. low pattern: v_l is uniformly distributed within $[1, ref_l]$. Let this number of required visits be denoted as low_l.
3. high pattern: v_l is uniformly distributed within $[ref_l, |F_l|]$. Let this number of required visits be denoted as $high_l$.
4. random pattern: v_l is uniformly distributed within $[low_l, high_l]$

As a result, from the 32 original instances (24 symmetric + 8 asymmetric) the total number of F-CVRP instances generated was: 32×3 number of families $\times$ 4 visit patterns $= 384$ instances.

As will be later shown, the proposed methodology was successful in producing solutions very close to the upper bounds reported in the literature. In addition, it matched the best solution scores for the vast majority of the instances and improved the best-known solutions for the remaining ones. This indicates that a new, larger-scale test bed is required for assessing the effectiveness of new optimization methods proposed for the F-CVRP. To this end, we have constructed a new collection of large scale F-CVRP benchmark instances. Four node sets are constructed which include $|N| \in \{150, 200, 300, 400\}$. For each node set, two cost matrices were generated (one symmetric and one asymmetric). For the resulting eight graphs the instance construction procedure of Bernardino and Paias (2024) was employed. The capacity Q as well as the family demand $d_l\ \forall\ l \in L$ are randomly assigned to each node set. The number of vehicles K is determined by solving a Bin Packing Problem based on family demands and visit requirements. Thus, in total, 96 new instances are introduced (8 graphs $\times$ 3 number of families $\times$ 4 visit patterns). This new data set is available online at: `https://github.com/KostasToulis/F-CVRP_Analytical_Solutions`.

4.2. Algorithmic Parameters

Throughout the presentation of the algorithmic framework, several key parameters have been referenced. They obviously affect the behavior of the proposed optimization method. Table 2 provides a summary of these parameters, detailing their role within the algorithm. It also reports their default values used in the *standard configuration* of our method. These values have been selected after tuning experiments on the benchmark data set aimed at optimizing the algorithmic performance.

Component	Parameter	Description	Default Value
Cyclic Framework	$maxRestarts$	Maximum number of non-improving cycles	$2 * \|N\|$
	$memProb$	Probability of applying the PSC strategy to generate the new starting solution	0.8
GTS	$maxNonImprove$	Maximum number of non-improving local search iterations	$2 \cdot \|N\|^2 + 2000$
	$promiseIter$	Re-initialization period for the tabu arc cost labels	Randomly assigned at each GTS cycle between $[75, 150]$
	$inOutProb$	Probability of using the in-out exchange move operator	$0.5 * (1 - \frac{\|P\|}{\|N\|})$
	g	Probability of considering the penalized cost matrix	0.4
	λ	Arc cost penalization coefficient	0.125
	$topArcs$	Percentage of the most costly solution arcs to be penalized	4%
PCS	$poolSize$	Maximum number of F-CVRP routes in the AM	$5 \cdot K$
	$maxLen$	Maximum eligible chain length	Determined randomly in $[0.5 \cdot avgLen, 0.7 \cdot avgLen]$
SCS	$maxSub$	Maximum number of node substitutions for the MCS IP model	$\|N\| - \|P\|$
	τ	Customer Removal penalization cost coefficient	0.1

Table 2: Algorithmic Parameters

Parameters g, λ and $topArcs$ jointly affect the behavior of the objective function modification strategy within the GTS method. The g parameter controls the frequency of arc penalization and usage of the augmented costs. Parameter λ determines the arc cost penalization intensity. Preliminary experiments across a range of different λ values revealed only minor variations in algorithmic behavior, with 0.125 ultimately selected as the most effective setting. Finally, $topArcs$ determines the number of most costly arcs that are penalized every time the penalized cost matrix is used. Obviously, it

depends on the problem scale. The larger the node population, the more arcs should be penalized to effectively diversify the search process.

Parameter *inOutProb* denotes the probability of applying the In-Out exchange operator. This value is depends on the problem features. It has been set proportional to the number of customers not currently included in the solution, as detailed in Section 3.2.1. The last GTS parameter *promiseIter* determines the period of re-initializing the arc cost labels. Our experiments indicated that randomly assigning this period for each cycle yielded better results compared to a single *promiseIter* value throughout the entire cyclic framework. Thus, the *promiseIter* value is uniformly distributed within $[75, 150]$ for each algorithm cycle.

Regarding the solution construction procedures, the most effective approach was a combination of both the PSC and SSC strategies, with a stronger emphasis on PSC, as indicated by the best *memProb* observed (0.8). The PSC strategy relies on two key parameters: *poolSize*, which defines the size of the route population stored in the AM, and *maxLen*, which sets an upper bound on the maximum chain length extracted for building new solutions. Experimental findings suggest that a rather small population of five solutions yield a powerful algorithmic performance. Setting the value of *maxLen* within $[0.5 \cdot avgLen, 0.7 \cdot avgLen]$ ensures that a significant part of the reconstructed starting solution retains promising characteristics stored in the memory structure, and at the same time it permits new arcs to be introduced in the solution towards underexplored regions of the solution space. As for the SSC component, we allow the model to perform the maximum number of substitutions possible by setting *maxSub* equal to $|N| - |P|$. This enables drastic modifications on the reconstructed solution structure.

The parameters controlling the objective function modification strategy and the two solution construction methods play a crucial role in shaping the algorithm's behavior. Therefore, their tuning experiments are examined in greater detail in the following sections.

4.2.1. GLS Contribution

In the first steps of the algorithm development, the local search process relied solely on the tabu policy of 3.2.2, forming a standard Tabu Search (TS) framework that delivered satisfactory performance. However, further experimentation with a Guided Local Search (GLS) implementation demonstrated also a very effective performance. In specific, GLS appeared to be more powerful for larger scale instances, whereas the pure tabu method was stronger for the smaller ones. To capitalize on the strengths of both methods, we developed a hybrid approach that combines them through a probabilistic mechanism controlled by parameter g, which defines the probability of using the penalized cost matrix for move selections. Parameter *topArcs* offers an additional means of regulating the intensity of this mechanism. It determines the number of arcs selected to be penalized. The GLS contribution within this hybrid procedure can be seen in the tuning results of these two parameters.

To determine the optimal setting for parameters g and $topArcs$, we conducted a factorial experiment using a grid search approach. Each parameter varied within a range of sensible values, and all possible parameter value pairs were tested, to assess their impact on solution quality. This tuning process was carried out using the light configuration of the algorithm (described in Section 4.3). The performance of each parameter pair was benchmarked against the best-known results reported by Bernardino and Paias (2024) (obtained by their heuristic or MILP formulations). Figure 3 provides a heatmap summarizing the results of the parameter tuning experiment. It illustrates the percentage of instances for which our algorithm matched or improved the best-known solutions across all tested value pairs of g and $topArcs$. The first line of the heatmap ($g = 0$) is a pure tabu-search and thus the results are not dependent on $topArcs$. The best values for these parameters were found to be $g = 0.4$ and $topArcs = 4\%$, and obviously this is the final value pair used in the proposed F-CVRP methodology. An interesting observation empirically supporting the strength of our hybridization is that the algorithm performance seems weaker in the upper-left and lower-right part of the grid. These two regions correspond to a pure tabu-search and an intense GLS method, respectively.

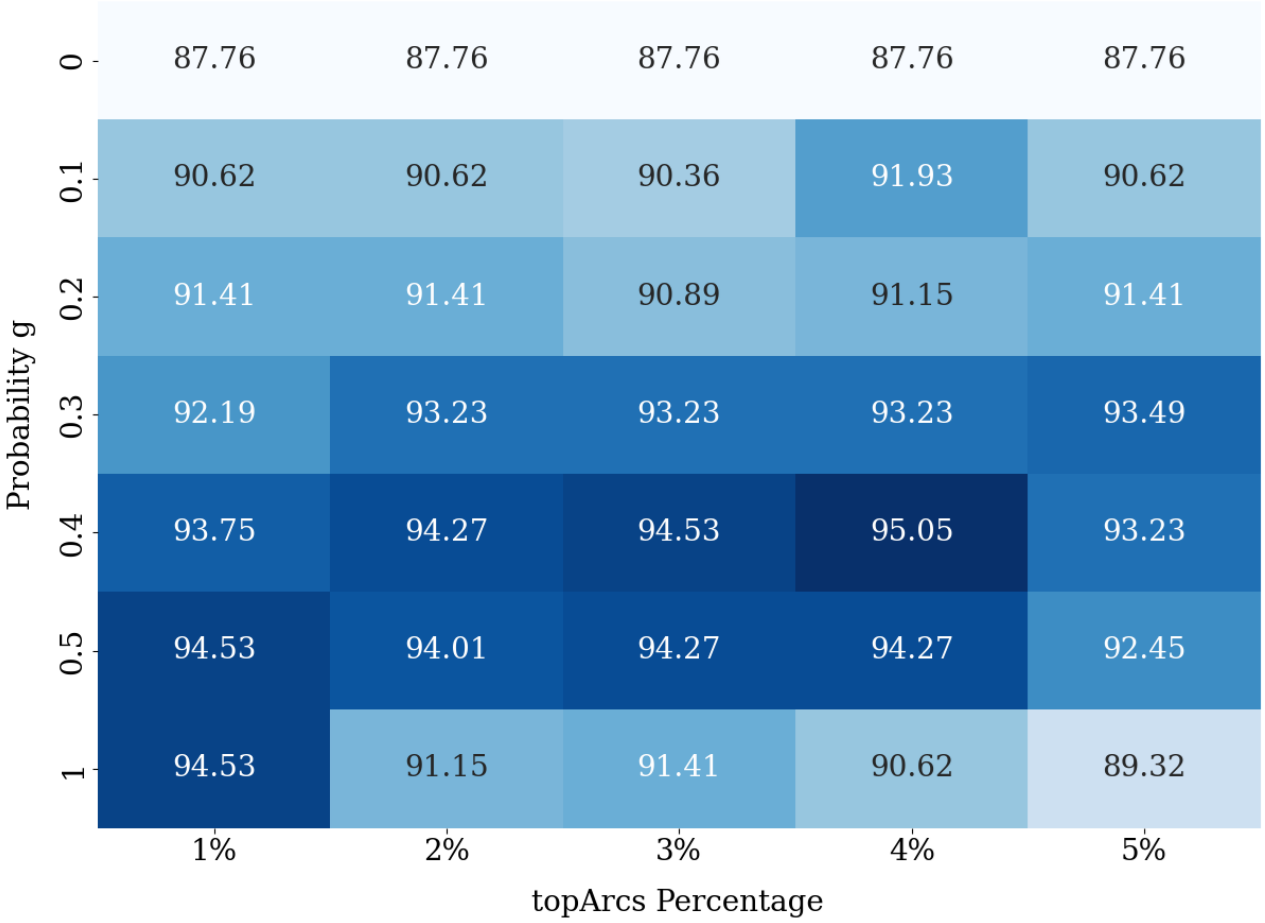

Figure 3: Percentage of instance solutions matched or improved by g, $topArcs$

4.2.2. The contribution of the Solution Construction Strategies

In the initial design of the algorithm, we explored two distinct approaches for constructing new solutions, namely PSC and SSC. They were described in Sections 3.3.1 and 3.3.2, respectively. The PSC procedure is based on Adaptive Memory Programming (AMP) which constitutes a well established metaheuristic framework with numerous successful applications in the VRP domain (Éric D Taillard et al. (2001), Zachariadis et al. (2010)). The SSC approach draws inspiration from existing methods that solve integer programming subproblems to construct new solutions by performing modifications to existing ones. These methods have demonstrated effective performance for other vehicle routing problems (Dontas et al. (2023), Manousakis et al. (2022)). Given their strength, we tested both strategies within the proposed cyclic optimization framework. When used separately, they produced similar results in terms of solution quality. We aimed to investigate whether there is any synergy between the two strategies. To this end, we introduced the parameter $memProb$, which controls the probability of selecting each strategy at every cycle of the proposed framework. $memProb$ took values from {0, 0.5, 0.6, 0.7, 0.8, 0.9, 1}. The light configuration of the proposed matheuristic was applied to the Bernardino and Paias (2024) instances for each tested $memProb$ value. The obtained results are graphically presented in Figure 4. It presents the percentage of the best-known solutions of Bernardino and Paias (2024) (obtained by their heuristic or MILP formulations) that were matched or improved. The results suggest that jointly using these strategies obtains better results compared to the individual use of PSC and SSC. The most effective combination of strategies occurs when $memProb = 0.8$, reflecting a bias in favor of the PSC strategy: while the PSC leverages the structure of high-quality solutions to intensify the search around promising regions, the SSC plays a complementary role by injecting diversification. It enables the algorithm to occasionally escape local optima and explore less-visited areas of the solution space, thus improving the overall search process.

4.3. Instance Results

To assess the effectiveness of the proposed methodology, we have executed the standard setting of our algorithm to the 384 benchmark instances of Bernardino and Paias (2024). The obtained results are compared against the best-known solutions reported in the literature. The work of Bernardino and Paias (2024) mainly focused on solving MILP F-CVRP formulations with the use of CPLEX 20.1. Secondary focus was given on heuristics, with the implementation of an Iterated Local Search (ILS) component which utilized a perturbation procedure to escape from local optima. In specific, Bernardino and Paias (2024) applied the heuristic strategy only to the symmetric instances for which the exact methods failed to reach optimality.

Our first benchmarking scheme is aimed at comparing the solutions obtained by the standard setting of our algorithm against the best-known solution obtained either by the exact or the heuristic method of Bernardino and Paias (2024). A maximum time limit of one hour has been given the

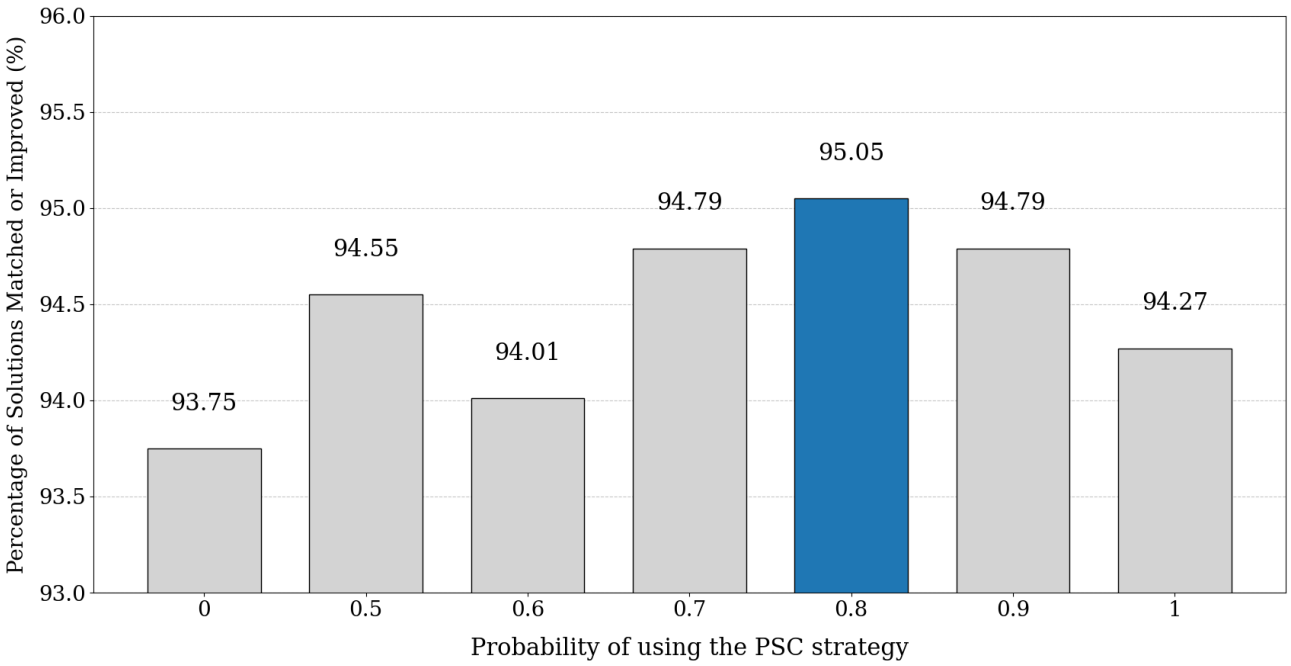

Figure 4: Tuning results of memProb

algorithm to produce the best solution for each instance. Table 3 summarizes the comparative analysis for the asymmetric and the symmetric instances, aggregated on the number of customers. Column $\#Solved$ indicates the number of instances for which our algorithm managed to match or improve the best solution. Column $\%\overline{Gap}$ reports the average percentage gap between our obtained score and the previously known best solution score, whereas $\overline{T}(s)$ represents the average CPU time in seconds. Note that this this represents the average time at which the algorithm completed its run, not the moment when the best solution was found. Detailed scores per instance are presented in the accompanying supplementary material document. The analytical solutions may be found online: `https://github.com/KostasToulis/F-CVRP_Analytical_Solutions`

Asymmetric				Symmetric			
$\|N\|$	$\#Solved$	$\%\overline{Gap}$	$\overline{T}(s)$	$\|N\|$	$\#Solved$	$\%\overline{Gap}$	$\overline{T}(s)$
33	12/12	0.0	15	15	12/12	0.0	0
35	12/12	0.0	21	18	12/12	0.0	1
38	12/12	0.0	27	19	12/12	0.0	1
44	12/12	0.0	59	20	12/12	0.0	2
47	12/12	0.0	84	21	24/24	0.0	2
55	12/12	0.0	155	22	12/12	0.0	1
64	12/12	0.0	506	39	12/12	0.0	25
70	12/12	0.0	581	44	12/12	0.0	48
				49	36/36	0.0	72
				50	12/12	-0.16	83
				54	48/48	-0.13	103
				59	24/24	-0.36	170
				64	12/12	-1.1	302
				69	12/12	-0.7	481
				75	24/24	-0.26	886
				100	12/12	-0.58	2980

Table 3: Benchmark instance results

The results highlight the effectiveness of the proposed matheuristic strategy. It has demonstrated robustness across a diverse set of instances, varying in the number of customers, families, and visit requirements. Within reasonable CPU time, the algorithm successfully matches the optimal solution in 337 of 384 instances and improves the best known upper bound in 47 of them, reporting an overall -0.13% average gap from previously published metrics. Bernardino and Paias (2024) noted that their MILP formulations had greater difficulty reaching optimal solutions for symmetric instances compared to asymmetric ones of similar size. On the contrary, our matheuristic framework appears to have a consistent behavior both for the symmetric and the asymmetric instances. As such, the results presented in Table 3 demonstrate a stronger dominance of our approach for the symmetric instances.

Our second benchmarking scheme is aimed at comparing the solutions obtained from our methodology against the heuristic results reported by Bernardino and Paias (2024) for the symmetric instances. To enable a fair comparison, particularly with respect to the computational effort, this benchmarking scheme is performed under a lighter version of our proposed solution methodology. The standard number of non-improving GTS iterations $maxNonImprove$, is reduced to $|N|^2+2000$ and the number

of non-improving cycles $maxRestarts$ to $\frac{|N|}{2}$. This lighter setting requires comparable CPU time with the heuristic of Bernardino and Paias (2024). In addition, since their heuristic involved probabilistic steps, the authors executed their algorithm five times on each benchmark instance. Our comparisons are made against the best solution scores obtained over those five runs.

The results obtained are summarized in Table 4. In this analysis, column $\#Solved$ indicates the number of instances for which our algorithm, using the lighter configuration, successfully matched the best known solution available. As reference, the best solutions in this comparison are the ones produced by our matheuristic optimization framework under the standard configuration. The column $\%\overline{Gap}$ reports the average percentage gap between the solution scores obtained and the best-known reference values. The average computational time is reported in column $\overline{T}(s)$. These three columns are repeated for the results of Bernardino and Paias (2024) obtained through their heuristic and are denoted as $\#Solved_B$, $\%\overline{Gap_B}$, $\overline{T_B}(s)$. Detailed solution scores per instance are available in the accompanying supplementary material document.

$\|N\|$	$\#Solved$	$\%\overline{Gap}$	$\overline{T}(s)$	$\#Solved_B$	$\%\overline{Gap_B}$	$\overline{T_B}(s)$
15	12/12	0.0	0	12/12	0.0	0
18	12/12	0.0	0	12/12	0.0	0
19	12/12	0.0	0	12/12	0.0	0
20	12/12	0.0	0	12/12	0.0	1
21	24/24	0.0	0	24/24	0.0	1
22	12/12	0.0	0	12/12	0.0	1
39	12/12	0.0	1	10/12	0.11	1
44	11/12	0.06	2	8/12	0.26	6
49	33/36	0.03	2	29/36	0.15	8
50	12/12	0.0	3	7/12	0.25	7
54	45/48	0.02	4	28/48	0.28	7
59	22/24	0.01	6	12/24	0.42	7
64	9/12	0.07	9	9/12	1.37	12
69	11/12	0.01	11	1/12	0.98	12
75	15/24	0.16	22	0/24	3.63	20
100	8/12	0.08	99	0/12	4.58	38

Table 4: Heuristic method comparison

Firstly, when evaluated against previously reported heuristic-derived results, our matheuristic approach shows a clear performance advantage under similar computational time. Notably, this improvement is even more pronounced in larger instances, where for $|N| \geq 75$ the previous heuristic fails to match the best-known solution in all instances. Overall, it matches a total of 71.88% of the

best-known solutions, compared to 93.23% of the proposed matheuristic in this light configuration. Exact runtime comparisons are not the focus of this analysis, as differences in hardware, programming language, implementation choices, and other factors can significantly affect execution time. Nonetheless, under the light configuration, the proposed matheuristic consistently delivers results within a comparable time frame to those reported in the literature.

From the presented results, it is evident that the original test bed has been thoroughly explored and that the obtained solutions are very close to the optimal values. Thus, we understood that the development of a more challenging data set, discussed in 4.1, was necessary to further assess this methodology and support future research efforts. To solve this new data set, a modified algorithm configuration was applied: the number of non-improving GTS iterations was set to $maxNonImprove = 2 \cdot |N|^2 + 2000$, capped at a maximum of 50.000. While no limit was placed on the total number of cycles ($maxRestarts = \infty$), each instance was given one hour to compute the best possible solution. Detailed scores are presented in the accompanying supplementary material document, and analytical solutions for these new instances can be found online: `https://github.com/KostasToulis/F-CVRP_Analytical_Solutions`

5. Conclusion

The F-CVRP is a novel variant of the Vehicle Routing Problem with real-life applications where customers are clustered into groups and each group has a predetermined number of required visits. One such application is in order-picking scenarios in large-scale warehouses, where multiple identical items of the same SKU are stored in different locations. Its structure reflects practical constraints, such as predefined picking requirements per SKU to fulfill incoming orders, making it highly relevant for modern logistics and distribution environments. The examined problem is defined by two key features: customer clustering and selective service. In light of these characteristics, an extensive review of the VRP literature was carried out to present related problem variants and the optimization methodologies that have proven to be most effective in tackling them.

The examined problem is solved via a matheuristic optimization methodology. The proposed methodology integrates several complementary optimization strategies to effectively address the complexity of the F-CVRP. It is a cyclic optimization framework, with a hybrid Guided Tabu Search (GTS) component acting as the central solution refinement mechanism. To prevent premature convergence and promote broader exploration of the solution space, a tabu policy with aspiration criteria is incorporated in the local search process. In addition, the objective function modification strategy as dictated by the Guided Local Search was seen to be an effective diversification mechanism, particularly for larger problem instances. Thus, our algorithm was developed as a hybridization of Tabu and Guided Local Search. The cyclic framework repetitively applies the Guided Tabu Search to starting solutions generated by two distinct strategies. The first one combines components of high-quality

solutions generated and stored through the search process, whereas the second one applies drastic solution modifications by solving a suitably defined MILP subproblem which calls for the optimal substitution of customer sets.

Computational results across the full set of 384 published benchmark instances demonstrate the effectiveness of our approach. These instances involve up to 100 customers. The algorithm successfully improved the best-known solutions for 47 instances, and consistently matched the best solution scores for the rest 337 test problems. To further support the research community and future research efforts, a new large-scale F-CVRP data set was introduced and solved. It features instances with up to 400 customers. This data set serves as a valuable benchmark for evaluating algorithmic performance on more realistic and demanding scenarios. In terms of future research paths, several hybrid MILP and metaheuristic approaches can be developed to target the interdependent decision layers of the problem, namely optimal customer selection and vehicle routing.

6. Acknowledgments

The research project was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "2nd Call for H.F.R.I. Research Projects to support Faculty Members & Researchers"(Project Number: 02562).

The authors would like to thank Dr. Eleftherios Manousakis and Mr. Vasileios Mylonas for offering constructive remarks during the algorithm development.

Appendix A. Instance Results

Detailed result tables are available in the accompanying supplementary document. Additionally, analytical solution scores both for the original test bed and the new F-CVRP instances are available online at: `https://github.com/KostasToulis/F-CVRP_Analytical_Solutions`

References

Angelelli, E., Archetti, C., and Vindigni, M. (2014). The clustered orienteering problem. *European Journal of Operational Research*, 238(2):404–414.

Archetti, C., Carrabs, F., and Cerulli, R. (2018). The set orienteering problem. *European Journal of Operational Research*, 267(1):264–272.

Archetti, C., Carrabs, F., Cerulli, R., and Laureana, F. (2024). A new formulation and a branch-and-cut algorithm for the set orienteering problem. *European Journal of Operational Research*, 314(2):446–465.

Archetti, C., Feillet, D., Hertz, A., and and, M. G. S. (2009). The capacitated team orienteering and profitable tour problems. *Journal of the Operational Research Society*, 60(6):831–842.

Augerat, P. (1995). *Approche polyèdrale du problème de tournées de véhicules*. PhD thesis, Institut National Polytechnique de Grenoble-INPG.

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

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

Boysen, N., de Koster, R., and Weidinger, F. (2019). Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, 277(2):396–411.

Butt, S. E. and Cavalier, T. M. (1994). A heuristic for the multiple tour maximum collection problem. *Computers Operations Research*, 21(1):101–111.

Chisman, J. A. (1975). The clustered traveling salesman problem. *Computers Operations Research*, 2(2):115–119.

Christofides, N. (1976). The vehicle routing problem. *Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle*, 10(V1):55–70.

Defryn, C. and Sörensen, K. (2017). A fast two-level variable neighborhood search for the clustered vehicle routing problem. *Computers Operations Research*, 83:78–94.

Dontas, M., Sideris, G., Manousakis, E. G., and Zachariadis, E. E. (2023). An adaptive memory matheuristic for the set orienteering problem. *European Journal of Operational Research*, 309(3):1010–1023.

Fischetti, M., Toth, P., and Vigo, D. (1994). A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. *Operations Research*, 42(5):846–859.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*.

Hintsch, T. (2021). Large multiple neighborhood search for the soft-clustered vehicle-routing problem. *Computers Operations Research*, 129:105132.

Hintsch, T. and Irnich, S. (2020). Exact solution of the soft-clustered vehicle-routing problem. *European Journal of Operational Research*, 280(1):164–178.

Manousakis, E. G., Kasapidis, G. A., Kiranoudis, C. T., and Zachariadis, E. E. (2022). An infeasible space exploring matheuristic for the production routing problem. *European Journal of Operational Research*, 298(2):478–495.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *J. ACM*, 7(4):326–329.

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

Nguyen, T. D., Martinelli, R., Pham, Q. A., and Hà, M. H. (2025). The set team orienteering problem. *European Journal of Operational Research*, 321(1):75–87.

Sevaux, M. and Sörensen, K. (2008). Hamiltonian paths in large clustered routing problems. *Proceedings of the EU/MEeting 2008 workshop on Metaheuristics for Logistics and Vehicle Routing, EU/ME*.

Speranza, M., Archetti, C., and Vigo, D. (2014). *Chapter 10: Vehicle Routing Problems with Profits*.

Tsiligirides, T. (1984). Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35(9):797–809.

Voudouris, C. and Tsang, E. (1996). Partial constraint satisfaction problems and guided local search. *Proc., Practical Application of Constraint Technology (PACT'96), London*, pages 337–356.

Zachariadis, E. and Kiranoudis, C. (2011). Local search for the undirected capacitated arc routing problem with profits. *European Journal of Operational Research*, 210(2):358–367.

Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T. (2010). An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries. *European Journal of Operational Research*, 202(2):401–411.

Zachariadis, E. E., Tarantilis, C. D., and Kiranoudis, C. T. (2015). The load-dependent vehicle routing problem and its pick-up and delivery extension. *Transportation Research Part B: Methodological*, 71:158–181.

Éric D Taillard, Gambardella, L. M., Gendreau, M., and Potvin, J.-Y. (2001). Adaptive memory programming: A unified view of metaheuristics. *European Journal of Operational Research*, 135(1):1–16.