Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi
Pamukkale University Journal of Engineering Sciences

A variable neighbourhood descent algorithm with tabu mechanism for the time-constrained family travelling salesman problem

Süre kısıtlı aile gezgin satıcı problemi için tabu mekanizmalı değişken komşu iniş algoritması

Beyza Günesen Akansu[1*]

[1]Department of Industrial Engineering, Faculty of Engineering and Architecture, Eskişehir Osmangazi University, Eskisehir, Türkiye.
beyza.gunesen@gmail.com

Received/Geliş Tarihi: 01.11.2024 Revision/Düzeltme Tarihi: 18.09.2025 doi: 10.65206/pajes.34901
Accepted/Kabul Tarihi: 06.10.2025 Research Article/Araştırma Makalesi

Abstract

In this study, the Family Travelling Salesman Problem is considered and time constraints are included in the model to better represent real-life applications. The mathematical model for the proposed problem has been adjusted as necessary and a metaheuristic method has been developed in order to achieve good solutions in shorter times. The method is a Variable Neighbour Descent algorithm using four different neighbourhood structures and a tabu list is added to the algorithm to be used in some neighbourhood movements to make the solution space search more efficient. The perturbation operator also diversifies the search by making large changes on the solution. The proposed algorithm was compared with the mathematical model results and performed better on the sample sets used.

Keywords: Family travelling salesman problem, variable neighbourhood descent, travelling salesman problem

Öz

Bu çalışmada, Aile Gezgin Satıcı Problemi ele alınmış ve gerçek yaşam uygulamalarını daha doğru yansıtabilmek için modele zaman kısıtları dahil edilmiştir. Önerilen problemin matematiksel modeli gerektiği şekilde uyarlanmış ve daha kısa sürelerde iyi çözümler elde edebilmek amacıyla bir metasezgisel yöntem geliştirilmiştir. Bu yöntem, dört farklı komşuluk yapısı kullanan ve bazı komşuluk hareketlerinde tabu listesi eklenerek çözüm uzayının daha verimli taranmasını sağlayan Değişken Komşu İniş algoritmasıdır. Ayrıca, çözüm üzerinde büyük değişiklikler yaparak aramayı çeşitlendiren bir pertürbasyon operatörü de uygulanmıştır. Önerilen algoritma, matematiksel modelin sonuçlarıyla karşılaştırılmış ve kullanılan örnek setlerinde daha iyi performans göstermiştir.

Anahtar: Aile gezgin satıcı problemi, değişken komşu iniş, gezgin satıcı problemi

1 Introduction

Routing problems have been widely studied in operations research due to their extensive real-life applications [1]. The common objective in such problems is to generate optimised routes that optimise an objective function, the most common objective is cost minimisation. The Travelling Salesman Problem (TSP) [2] is one of the most extensively researched routing problems. Given a starting node (depot), a set of cities and a matrix of costs between cities, the TSP aims to identify the least-cost Hamiltonian tour. Over the years since the problem's inception, numerous variants of the TSP representing real-life problems have been introduced in different studies. A variant of the Generalized Travelling Salesman Problem (GTSP) arises when the visiting nodes are divided into clusters and exactly one node from each cluster must be visited [3]. This is known as the Family Travelling Salesman Problem (FTSP). However, this problem includes the condition that not only one node from each set of nodes must be visited, but a predetermined number of nodes.

Generalized Covering Salesman Problem (GCSP), defined by [4] is another problem that is similar to FTSP. The objective of the GCSP is to identify the optimal route, in terms of cost, that encompasses all specified nodes. The primary characteristic of this problem is that each data node is covered by at least one route node, subject to the coverage constraint. In GCSP, the nodes that fulfil the coverage constraint with each other are in the same cluster, while in FTSP they are in the same family. There are many studies in the literature for these problems [5],[6].

The concept of FTSP was first introduced by [7] to describe the order picking problem in warehouses where products of the same type are stored in different locations, whether in different warehouses or within the same warehouse. Considering the recent technological developments, it is no longer necessary to store the same products in the same departments. Warehouses that adopt a chaotic storage system [8], where there are no predetermined storage sections for storing products, are the practical applications of FTSP. Such storage systems allow for more flexible optimisation of order tracking and management. In GCSP, the nodes are represented as clusters and all clusters must be visited, in which case, if the clusters are considered as single elements, the problem becomes a TSP and is classified as an NP-hard problem like TSP [6]. FTSP is also a special subtype of GCSP and is in the same difficulty level. Due to the NP-hard nature of the problem, the use of heuristic algorithms becomes inevitable as the problem size increases.

This study addresses FTSP, a topic that has received less attention than similar issues in the existing literature. The study's main contributions are outlined below.

*Corresponding author/Yazışılan Yazar

- A new mathematical model has been devised by incorporating the times for receiving and loading the stored products into the practical applications of FTSP.
- New test problems were generated by incorporating the requisite parameters for the novel model into the existing FTSP test problems documented in the literature.
- A variable neighbour search-based meta-heuristic has been developed to address the problem dimensions where the mathematical model is insufficient.

The second section reviews the existing literature on FTSP, followed by the third section, which introduces the developed model. The fourth section explains the proposed metaheuristic algorithm, and the final section provides a comparative analysis of the model and metaheuristic results.

2 Literature rewiev

The objective of the literature review was to examine how the FTSP has been addressed and the approaches that have been adopted for solutions. Given that the FTSP is a novel variant of the TSP, it has not yet been sufficiently examined in the existing literature.

[7] introduced the FTSP for the management of warehouses using modern technologies such as radio frequency identification (RFID) where similar products are not stored together, concluding that the TSP, which has been used to model the warehouse management problem, is inadequate. A binary integer mathematical programming model was developed for the FTSP. Two metaheuristic algorithms, the Biased Random-Key Genetic Algorithm (BRKGA) [9] and the Greedy Randomized Adaptive Search Procedure (GRASP) with Evolutionary Path Relinking (GRASP+evPR) [10], are proposed for larger-size problems where the CPLEX solver is not sufficiently effective. To assess the efficacy of the proposed methodologies, seven instances from the TSPLIB library [11] have been adapted to the FTSP domain, and a total of 21 test instances have been formulated. The results demonstrate that both algorithms exhibit promising performance on large-scale instances, with BRGA exhibiting a notable advantage over GRASP+ evPR on both large and medium-sized problem sets.

[12] proposed mixed-integer models for FTSP that differ in subtour elimination constraints. The models are classified into two categories: compact and non-compact. Compact models comprise flow variables, whereas non-compact models include cut inequalities. The results demonstrate that non-compact models are more time-efficient. Furthermore, the authors propose an Iterative Local Search (ILS) approach for large-scale instances of FTSP and develop best known upper bounds.

[13] proposed a hybrid algorithm, Iterative Local Search (ILS) and Genetic Algorithm (GA) [14] metaheuristics that incorporate branch-and-cut and local search. In [12], two novel neighbourhood mechanisms and a solution perturbation operator were incorporated into the proposed ILS. The two new metaheuristics enhanced the optimal upper bounds documented in the literature for the problem. In addition to the test instances, [7] devised a test instance generator based on the cost matrices employed in TSP test instances in the literature and presented new FTSP test instances.

[15] proposed an Incompatibility Constrained FTSP model that considers the constraint of nodes in the same family that are incompatible with each other. This constraint provides the conditions that members of the same family should not be on the same route. Incompatibility constraints are modelled by defining appropriate graphs for each family. Compact and non-compact mixed-integer mathematical models are developed and optimal solutions are obtained up to sample sets consisting of 127 nodes by branch-and-bound method. Two metaheuristics, ILS and Ant Colony Optimisation (ACO) [16], were developed and optimal solutions for the new problem were obtained in shorter times than the exact method.

[17] introduced clustered FTSP to model warehouse systems with scattered storages. The authors developed mixed-integer models for multi-depot FTSP, hard clustered multi-depot FTSP, and soft clustered multi-depot FTSP. Branch-and-cut based algorithms were developed for solving the models, and the validity of the models was tested using instance sets comprising up to 200 nodes and 40 depots. The differences between the three new proposed problems are presented, and it is found that the hard clustered multi-depot FTSP problem is a more challenging problem in terms of computational efficiency.

[18] focused on solving FTSP with the DNA computational Adleman-Lipton model [19],[20]. DNA computing seeks to solve the problem by mimicking biological processes applied to DNA molecules. Due to the parallel processing capabilities of DNA computing, it is stated that such problems can be solved more efficiently compared to classical digital computing.

In the matheuristic approach proposed by [21], mathematical programming is used for the local search process in different parts of the solution and Genetic Algorithm is used for the combination of solution parts. The generation of feasible solutions was conducted using a GA, while the selection between feasible solutions was performed using an Simulated Annealing (SA) [22]. Subsequently, the solutions are enhanced through a procedure devised in accordance with the concept of Partial Optimisation Metaheuristic Under Special Intensification Conditions (POPMUSIC) [23]. This improvement is conducted on a part of the solution each time by applying the mathematical programming model.

[24] defined the Capacitated FTSP (CFTSP), which differs from the classical FTSP in that it has more than one agent and takes into account the capacities of the agents. They stated that capacitated agents better reflect the real-life problems of order picking from warehouses. They proposed integer linear programming models with five different subtour inequalities. A Biased Random-Key Genetic Algorithm (BRKGA) with four different decoding approaches was developed to produce quality solutions in a short time.

The first algorithm proposed by [25] for FTSP is the Parallel Branch Cut Method (P-BC), where the parallel processing capability is provided by local search. The branch-and-bound component is responsible for ensuring the optimality of the route and identifying interesting node clusters on the same route. The second component comprises new node search strategies that take these clusters into account, thereby enabling rapid pruning of nodes in the branch-and-bound tree and local search. Another method proposed for the problem is the Biased Random Key Genetic Algorithm with Q- Learning (BRKGA-Q) [26], which combines metaheuristic and machine learning techniques.

The Q-learning algorithm provides control of the parameters in the evolutionary processes in BRKGA. P-BC reached optimal solutions in the vast majority of known test cases and approximately 75% of the test instances with unknown optimal values. The metaheuristic method obtained the best known solutions in a reasonable time for large-scale problems.

[27] developed a hyperheuristic method in which three different Large Neighbourhood Search (LNS) [28] algorithms are represented as low-level heuristics, taking into account the

subset selection and permutation properties of the problem. In the hyperheuristic method, the selection of low-level heuristics is either greedy or randomized. In computations utilizing existing test cases, the proposed approach obtained optimal solutions in a shorter time than existing methods. Furthermore, new test instances were introduced for the problem, and the hyperheuristic technique demonstrated satisfactory performance on these cases as well.

The FTSP has been addressed through exact models and a variety of metaheuristic approaches, including ILS, GA, BRKGA, ACO, and hybrid methods. Several extensions such as incompatibility-constrained, clustered, and capacitated FTSP have also been studied. Overall, the literature shows that while important progress has been achieved, further research is needed to develop efficient algorithms for large-scale and real-world applications.

3 Problem definition and mathematical model

The Family Travelling Salesman Problem (FTSP) has the properties that nodes in the same family are located in different locations and a certain number of nodes from each family must be visited. This problem is introduced in order to ensure that the products are collected with minimum cost in warehouses where RFID system is used. In this study, it is aimed to increase the similarity of the problem to real life situations by including the time taken during the transfer of the products in the model. In cases where products must be collected within a certain period of time, or where the energy source (charge, fuel, battery percentage, etc.) of the material handling system responsible for product collection and unloading is limited, it may become necessary to collect products within a certain time frame. Moreover, these times may vary for each product.

FTSP can be represented by a fully connected graph, denoted by G(V,A). $V = \{0,1,2,\dots,N\}$ is a set of nodes consisting of N+1 nodes, where node 0 is the starting node, i.e. the depot.

Sets	Explanation
$\boldsymbol{V}$	Set of nodes including depot. $V = N \cup 0$
$\boldsymbol{N}$	Set of nodes.
$\boldsymbol{A}$	Set of edges $A = \{(i,j) \mid i,j \in V\}$
$\boldsymbol{L}$	Set of families $L = \{1,2,\dots,f\}$

Parameters	Explanation
$\boldsymbol{f}$	Number of families
$\boldsymbol{E_l}$	Number of nodes (members) from family l; $l \in \{1,2,\dots,f\}$
$\boldsymbol{Z_l}$	Number of nodes (members) from family l which must be visited
$\boldsymbol{KN}$	Total number of nodes to be visited; $KN = \sum_{l=1}^{L} z_l$
$\boldsymbol{F_1, F_2, F_3, \dots, F_f}$	Partitioned sets corresponding to families; $F_1 \cup F_2 \cup F_3 \cup \dots F_f \cup \{0\} = V$
$\boldsymbol{c_{ij}}$	Cost of edge $A \in (i,j)$
$\boldsymbol{t_i}$	Service (processing) time of node i
$\boldsymbol{T}$	Available total service time

Variables	Explanation
$\boldsymbol{x_{ij}}$	Binary decision variable that equals 1 if edge (i,j) is included in the solution
$\boldsymbol{u_i}$	Continuous auxiliary variable used in Miller–Tucker–Zemlin (MTZ) subtour elimination constraints to track node positions in the tour.

$$Min\ z = \sum_{(i,j)\in A} x_{ij} c_{ij} \tag{1}$$

$$\sum_{j\in N} x_{0j} = 1 \tag{2}$$

$$\sum_{i\in N} x_{i0} = 1 \tag{3}$$

$$\sum_{j\in V} x_{ij} \le 1 \qquad \forall_i \in N \tag{4}$$

$$\sum_{i\in V} x_{ij} \le 1 \qquad \forall_j \in N \tag{5}$$

$$\sum_{i\in V}\sum_{j\in V} x_{ij} \le KN + 1 \tag{6}$$

$$\sum_{i\in F_l}\sum_{j\in V} x_{ij} = Z_l \qquad l = \{1,2,\dots,f\} \tag{7}$$

$$\sum_{j\in F_l}\sum_{i\in V} x_{ij} = Z_l \qquad l = \{1,2,\dots,f\} \tag{8}$$

$$\sum_{(i,j)\in A} t_j x_{ij} \le T \tag{9}$$

$$u_i - u_j + 1 \le (n-1)(1 - x_{ij}) \qquad \forall_{i,j\in V} \tag{10}$$

$$x_{ij} \in \{0,1\} \qquad \forall_{i,j\in V} \tag{11}$$

$$u_i \in IR \qquad \forall_{i\in V} \tag{12}$$

Equation (1), which expresses the objective function, ensures that the distance or cost is minimised. Constraints (2) and (3) define that the tour should start from the depot and end at the depot. The constraints that ensure that a node is visited only once and then left are expressed by constraints (4) and (5). The sum of the number of visits in all families determines the total number of nodes that need to be visited and this is defined by constraint (6). Constraints (7) and (8) ensure that the required number of nodes from each family are visited and left. In contrast to the models presented in previous studies, constraint (9) considers the total processing time of the nodes to be visited in order to ensure that the total time available is not exceeded. (10) represent the subtour elimination constraints, also known as Miller-Tucker-Zemlin constraints [29]. (11) and (12) describe the properties of the decision variables. The mathematical model is based on the model proposed by [7], with the exception of (9), which prevents time violations, and (10), which prevents subtours.

Figure 1. shows a feasible solution for the instance Bayg_29. The points scattered on the coordinate plane represent the nodes to be visited. Nodes of the same family are visualized with the same color. According to the constraints in the mathematical model of the problem, given the number of members in each family and the number of nodes to visit from that family, the shortest possible route between the nodes to be

visited is determined without exceeding the maximum time available.

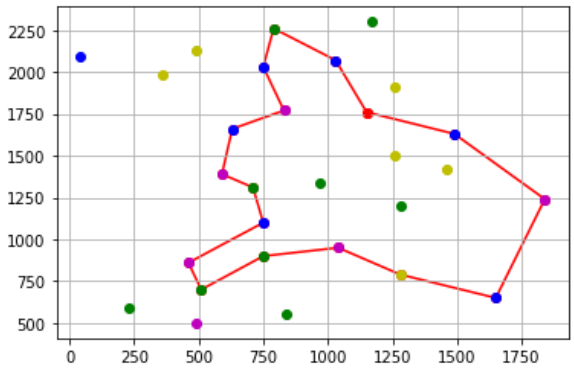

Figure 1. A feasible solution for Bayg_29.

4 Methodology

In TSPs, which are easy to define but very difficult to solve, the increase in the number of cities makes it difficult to find a solution in an acceptable time period with exact solution methods. Due to this feature, heuristic and metaheuristic methods are frequently used to solve the TSP, which is classified as NP-Hard [30]. Metaheuristic algorithms can find optimal solutions in a short time [31]. Since FTSP is in the NP-Hard problem class, a metaheuristic algorithm is needed to reach efficient solutions in a short time, so a Variable Neighbour Descent (VND) algorithm [32], which includes Tabu Search and perturbation mechanisms, is proposed in this study. VND is a local search algorithm that starts with an initial solution and improves the solution by iteratively searching between different neighbourhoods. While the VND algorithm allows for intensification during the search, the Tabu Search mechanism is embedded in the proposed algorithm in a similar approach to [33] in order to provide diversification. Thus, it is aimed to perform a more efficient search among the solutions. Tabu Search (TS) is a memory-oriented search method proposed by [34]. Unlike Local Search algorithms, it is an important method that has been used by researchers for years due to its high performance in exploring different regions of the solution space thanks to its memory feature. In the proposed method, diversification is achieved by avoiding the same similar solutions by preventing the repetition of the movements made in some of the neighbourhoods used in the proposed method during the tabu tenure. Diversification is enhanced by the tabu mechanism and a perturbation operator. Algorithm 1 contains the pseudocode of the proposed algorithm.

Algorithm 1: VND with Tabu Tool

```
Input: Max_Time, Max_Iter, NoImproveLimit, Tabu_Tenure, Tabu_List = {}
Output: Global Best Solution
Current ← Create_Initial_Solution();
Global_Best ← Current
no_improve_counter = 0
For iter = 1 to max_iter do
best_solution = None
      For all neighbourhoods
         new_solution ← Neighbourhood(current);
            if new_solution(time)< max_time and new_solution(cost)≤best_solution(cost) then
            best_solution←new_solution;
         if best_solution<current
            current←best_solution;
            Update(Tabu_list);
            Update(no_improve_counter);
         if current < global_best
            global_ best ← Local_Search(current);
         if no_improve_count greater than NoImproveLimit
            current ← perturbation(current);
Return global_best
```

Figure 2. Pseudecode for proposed algorithm.

The algorithm starts with an initial solution. The initial solution is a feasible solution obtained by running the mathematical model for a specific period of time for small-sized problems. For medium and large size problems, the initial solution is determined as a random feasible solution since a feasible solution cannot be found in a short period of time with the exact solution methods of the mathematical model. The max_sure in the input parameters expresses the available time constraint of the problem. Tabu list and tabu tenure are the elements of the tabu search mechanism defined as parameters and used in the algorithm. If the algorithm fails to improve for a certain number of iterations, this is a stopping condition and as a result the algorithm stops. If no improvement can be achieved for a smaller number of iterations before the stopping condition is reached, the perturbation mechanism is activated and a different region of the solution space is searched. VND allows iterative search between neighbourhoods, where four different neighbourhood structures are defined in the proposed algorithm. Based on the current solution, a certain number of neighbouring solutions are generated from a neighbourhood structure and saved as new_solution.

The best solution among the new solutions generated with the relevant neighbourhood structure is saved as best_solution. Best_solution is assigned as the current solution if it has a better result than the current solution. If the neighbourhood structure is one of the neighbourhood structures to be included in the tabu list, the tabu list is updated and the global_best check is performed. The loop is completed for all neighbourhoods and if there is no change in global_best during

a certain number of iterations, the perturbation operator is called and the current solution is modified and the algorithm continues over the new current solution. When a certain number of iterations is reached, the algorithm stops and returns the best solution found.

Neighbourhood structures determine the strategy for exploring the solution space and have a significant impact on the performance of the algorithm. The proposed VND includes four different neighbourhood procedures. The first neighbourhood structure is given in Algorithm 2 a random node from the current route is selected. The family of this node is identified, and then another node from the same family that has not yet been visited is randomly chosen. The visited node is removed from the route, and the unvisited family member is inserted into every possible position in the route. Finally, the best position is selected based on the minimum route cost.

The second neighbourhood given in Algorithm 3 procedure involves a similar approach to the first neighbourhood. While in the first neighbourhood a random node is first selected from the current route, in the second neighbourhood a random family is selected. A member of the selected family that is present in the current route is replaced by any of the remaining unvisited members of the family. After the replacement process, the newly added member is placed in the lowest cost position in the current route.

Neighbourhood 3 and Neighbourhood 4 structures are designed based on the LNS_1 and LNS_2 methods proposed by [27]. In their study, Pandiri and Singh [27] proposed a hyper-heuristic that selects between Large Neighbourhood algorithms. Among the algorithms, which they call LNS, selections are made according to different criteria. In the method proposed in this study, these search methods are added to the LNS algorithm as neighbourhood.

Algorithm 2: First Neighborhood Procedure

Input: A feasible solution, best_distance = ∞
Output: New solution, move
element1 ← **Choose** a random element from solution;
Family[i] ← **Find** family index for element1;
element2 ← **Choose** a random unvisited member from Family[i];
Extract (element1) from solution;
For each (index) in solution;
New_solution ← solution(element2)
New_distance ← calculate cost for new solution
if new_distance < best_distance **then**;
best_distance ← new_distance
best_location ← [index]
move ← (element1, element2)
Return New_solution, move

Figure 3. Pseudecode for neighbourhood-1.

Algorithm 3: Second Neighborhood Procedure

Input: A feasible solution, best_distance = ∞
Output: New solution, move
Family[i] ← **Choose** random family index for element1;
element1 ← **Chooese** a random visited member from Family[i];
element2 ← **Choose** a random unvisited member from Family[i];
Extract (element1) from solution;
For each (index) in solution;
New_solution ← solution(element2)
New_distance ← calculate cost for new solution
if new_distance < best_distance **then**;
best_distance ← new_distance
best_location ← [index]
move ← (element1, element2)
Return New_solution, move

Figure 4. Pseudecode for neighbourhood-2.

Algorithm 4: Third Neighborhood Procedure

Input: A feasible solution, best_distance = ∞
Output: New solution

```
n ← number of elements of feasible solution
Min_removed ← 1
Max_removed ← n
Num_removed←random number(min_removed,max_removed)
Removed_elements ← Extract num_removed number of elements from feasible solution and Add;

New_solution ← Extract Removed_elements from feasible solution;
For each elements in Removed_elements;
   New_solution ←Add element[i] best position in New_solution;
Return New_solution
```

Figure 5. Pseudecode for neighbourhood-3.

Algorithm 5: Fourth Neighborhood Procedure

Input: A feasible solution, best_distance = ∞
Output: New solution

```
n ← number of elements of feasible solution
Min_removed ← 1
Max_removed ← n
Num_removed←random number(min_removed,max_removed)
Removed_elements ← Extract num_removed number of elements from feasible solution and Add;
New_solution ← Extract removed_Elements from feasible solution;
For each element in removed_elements;
    Family [i] ←Find family index for element
    For unvisited_elements in Family[i];
        New_solution ←Add unvisited_element[i] best position in new_solution
        New_distance ← calculate cost for new solution
            if new_distance < best_distance then;
              best_distance ← new_distance
Return New_solution
```

Figure 6. Pseudecode for neighbourhood-4.

Neighbourhood 1 and neighbourhood 2 procedures involve low levels of perturbation and repair by replacing one node on the feasible solution. In Neighbourhood 3 and 4, a random number of nodes between 1 and the number of nodes in the route (n) are replaced, so the level of perturbation in the route is higher. For Neighbourhood 3, each element removed from the route is inserted to its best position in the route, i.e. the position that will provide the lowest cost. Neighbourhood 4, on the other hand, allows a large-scale change and the family indices of the nodes to be removed from the route are determined and each unvisited member within these families is placed in the best position in the route. Among the unvisited members, the member with the best cost value is selected and added to the best position sequence. Algorithms 4 and 5 show the pseudo-codes of neighbourhood 3 and neighbourhood 4 structures respectively.

Neighbourhood 1 and neighbourhood 2 allow to change a single node on the current route and this movement does not cause a significant change in the solution. In order to prevent these neighbourhoods from making the same movements over and over again and to scan the solution space more efficiently, their movements are stored in the tabu memory and the same movement is prevented from repeating for a certain period of time.

Within the scope of the algorithm, a better solution is obtained with neighbourhood functions, but this solution may not be locally best. If the new solution obtained in one iteration is better than the current solution, it is assigned as the current solution and the 2-opt local search algorithm can be applied to create a better route by correcting unnecessary long routes in the route and creating a shorter route. 2-opt creates a shorter route by selecting two edges in the current route and rearranging (reversing) these edges. If the distance between two nodes can be improved, these two edges are swapped, thus reducing the total distance.

The perturbation operator is used when the solution cannot be improved for a given number of iterations. The operator takes a sequence of the current route and inverts it. Perturbation allows the algorithm to explore different solution spaces. When all improvement opportunities in a given region are exhausted in local searches, the perturbation operator is used to move to a different region of the solution space.

In this study, computational experiments were carried out on benchmark instances of three different sizes to investigate the impact of the proposed neighborhood structures, both individually and in combination, on solution performance.

To ensure a fair comparison, the total number of candidate solutions generated in each iteration was kept constant. The strategies were defined according to the specific combinations

of neighborhoods employed. Here, N1, N2, N3, and N4 correspond to Neighborhood–1, Neighborhood–2, Neighborhood–3, and Neighborhood–4, respectively. Each strategy was evaluated over 10 independent runs for each test instance, and the minimum and average solution costs obtained were recorded.

Table 1. comparison of neighborhood strategies

Instance	bier_127_1 T = 173		a_280_1 T = 480		pr_264_1 T = 380	
Strategy	Min	Avg	Min	Avg	Min	Avg
N1	45989.70	75953.81	2035.29	2079.66	36815.0	39126.4
N2	46528.82	76007.72	2066.05	2104.93	37383.0	39392.6
N3	64470.82	78838.47	2125.33	2180.64	39242.0	41710.5
N4	64260.79	78200.18	2261.97	2339.27	40592.0	44010.4
N1→N3	43273.20	75682.16	1942.88	1996.02	35727.0	36747.2
N2→N3	47881.32	76142.97	1911.63	2012.48	36590.0	38092.2
N2→N4	49625.90	76317.43	2081.08	2152.13	36487.0	39415.5
N1→N4	50375.86	76392.42	2038.49	2094.55	37723.0	40093.9
N2→N1→N3	42225.81	75577.42	1918.76	2002.83	35572.0	36699.3
N1→N3→N4	42501.39	75604.97	1917.89	2013.62	36087.0	37822.6
N2→N1→N4	47031.97	76058.03	1919.96	2002.03	36748.0	38746.2
N2→N3→N4	48440.09	76198.84	1982.61	2041.81	37946.0	39268.3
N2→N1→N3→N4	41931.75	75548.01	1878.50	1961.29	35483.0	36668.5

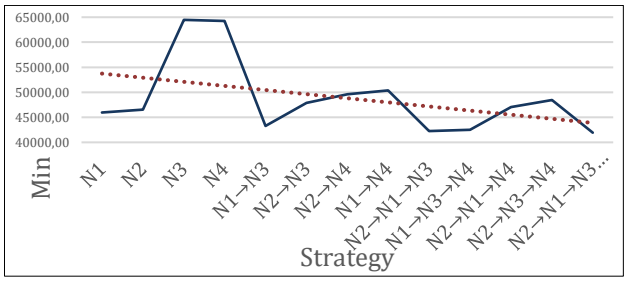

Figure 7. Trend of minimum solution values across neighborhood strategies

According to the results presented in Table 1 and Figure 7, the performance of different neighborhood strategies varies notably depending on whether they are applied individually or in combination. Single and paired neighborhoods yield relatively weaker results, while three-neighborhood strategies achieve moderate improvements. However, the four-neighborhood strategy (N2→N1→N3→N4) consistently outperforms the others across all benchmark instances, delivering the lowest minimum and average solution costs. These findings highlight that the proposed four-neighborhood strategy is the most effective approach, as the diversity of neighborhood structures significantly enhances the performance of the VND algorithm.

5 Computational experiments

In this study, which also considers the time constraint for FTSP, the instances first defined in the literature by [7] and recently added to the literature by [27] are used as a basis for comparing the mathematical programming model and algorithm and measuring their performance. Time parameters were added to some of the instance sets taken from these two studies, and they were made appropriate for the defined model.

All computational experiments were performed on a PC with Intel Core™ i5-8400 processor, 2.80GHz and 16 GB RAM. The algorithm and mathematical model were written in PYTHON version 3.8 and GUROBI 9.1.1 solver was used to solve the model.

In the first set of test instances shared by [7], the problem size ranged from 14 to 280. Each example was solved three times with different time constraint (T) values. T values were determined to be smaller than the total time value obtained in the solution without time constraint. The nodes to be visited are selected and sorted in such a way that the family member constraints are met, the maximum time available, i.e. the T value, is not exceeded and the total cost is minimised. Table 1 summarises the results of the province data set group. The GUROBI results of the model for different T values and the results of the proposed heuristic for the problem are presented in the table. In order to prove the time efficiency of the method, the solution time results of the method are given in seconds in the Time column. The method is run ten times for each instance and the results are shared with maximum, minimum and average values.

The model solution time was set to 120 seconds for results marked with (*) and 240 seconds for results marked with (**). In all problems where the model can reach the best solution, the proposed method has achieved the best results in minimum and average values. The proposed method showed better performance in all criteria in the results that the model could reach within the specified time.

The solutions obtained with the instances taken from the data sets shared by [27] in their study are given in Table 2. For these instance sets, the mathematical solution time was set as 300 seconds and no optimal solution was obtained in the specified time for any problem. The proposed VND algorithm with tabu mechanism achieved better results in much shorter times in all instances. In order to test the significance of the differences between the objective functions of the mathematical model and the proposed method, a statistical t-test was applied on the average results of the proposed method. Before the t-test, normality test was performed on both sets of results and it was determined that the values were normally distributed.

The null hypothesis was designed to state that there is no difference between the averages of both samples and the alternative hypothesis was designed to state that the average of the values including the method results is less. The confidence level for the t-test was set to 95%. If the p-value is less than 0.05 as a result of the analysis performed at the specified confidence interval, it will be concluded that the alternative hypothesis should be accepted, that is, it is statistically ensured that the metaheuristic method has better values according to the objective function. As a result of the analysis, p-value = 0.023 is found and since it is less than the threshold value, the method has lower total tour cost values.

Table 2. Results for first instance set.

	Gurobi	Metaheuristic				Gurobi	Metaheuristic				Gurobi	Metaheuristic			
Instance		Min	Max	Avr	Time		Min	Max	Avr	Time		Min	Max	Avr	Tine
			T = 8					T = 9					T = 10		
burma_14_1	21.32580	21.32580	21.32580	21.32580	0.16	15.73500	15.73500	15.73500	15.73500	0.16	13.9323	13.9323	13.9323	13.9323	0.15
			T = 30					T = 27					T = 25		
burma_14_2	25.6562	25.6562	25.6562	25.6562	0.28	25.8494	25.8494	25.8494	25.8494	0.29	25.8494	25.8494	25.8494	25.8494	0.29
			T = 5					T = 6					T = 7		
burma_14_3	19.9526	19.9526	19.9526	19.9526	0.32	14.3618	14.3618	14.3618	14.3618	0.44	11.8860	11.8860	11.8860	11.8860	0.50
			T= 42					T= 37					T = 34		
bayg_29_1	5345.86	5345.86	5345.86	5345.86	0.56	5366.08	5366.08	5366.08	5366.08	1.28	5394.01	5394.01	5394.01	5394.01	1.32
			T= 42					T = 40					T = 38		
bayg_29_2	5791.01	5791.01	5791.01	5791.01	1.21	5813.39	5813.39	5813.39	5813.39	1.81	5863.23	5863.23	5863.23	5863.23	1.75
			T= 44					T = 41					T = 36		
bayg_29_3	5544.33	5544.33	5544.33	5544.33	2.77	5775.91	5775.91	5775.91	5775.91	1.41	6171.53	6171.53	6171.53	6171.53	1.02
			T= 100					T=90					T = 86		
att_48_1	23686.02	23686.02	23686.02	23686.02	13.27	25584.55	25584.55	25584.55	25584.55	13.39	27639.26	27639.26	27639.26	27639.26	7.8
			T= 74					T = 70					T = 65		
att_48_2	20826.23*	20609.08	20826.23	20646.75	12.18	21232.54*	20653	20991	20754	12.2	20950.43*	20812.61	21246.49	20930.29	12.19
			T= 47					T = 40					T = 35		
att_48_3	9335.07*	9024.58	9100.79	9044.49	11.11	9292.32	9292.32	9353.1	9347.02	20.9	10482.19*	10420.58	10482.19	10463.7	23.2
			T= 173					T = 165					T = 160		
bier_127_1	44911.4*	37946.62	39256.71	38573.10	98.30	47072.4822*	38129.81	39935.12	39147.11	95.87	48465.92*	38423.47	50985.20	42297.82	98.4
			T= 245					T= 230					T = 210		
bier_127_2	105539.66*	90005.35	92751.26	91876.98	135.60	106276.36*	92668.14	94753	93910	144.0	113974.212*	97586.75	100619.82	99157.8	149.8
			T = 182					T= 170					T=160		
bier_127_3	53835.46*	47827.02	48557.07	48312.03	80.02	59381.88*	48698.96	49767.74	48900.25	80.29	56256.19*	49993.43	51468.62	50525.75	79.9
			T= 520					T = 500					T = 480		
a_280_1	3229.11**	1902.99	2376.16	1979.85	142.88	3323.622**	1814.82	1952.65	1891.82	202.19	3100.35**	1903.86	2040.91	1957.22	190.2
			T=443					T=430					T= 410		
a_280_2	3110.5**	1679.10	1775.66	1726.52	153.19	2998.08**	1699.32	1850.44	1748.49	151.2	2960.45**	1741.57	1832.20	1776.39	127.7
			T = 403					T= 390					T = 370		
a_280_3	2620.97**	1504.73	1631.23	1583.91	106.04	2826.9**	1500.3	1668.48	1598.15	102	2779.29**	1571.26	1709.8	1637.56	107.6

Table 3. Results for second instance set.

	Gurobi	Matheuristic				Gurobi	Matheuristic				Gurobi	Matheuristic			
Instance		Min	Max	Avr	Time		Min	Max	Avr	Time		Min	Max	Avr	Time
			T = 273					T = 260					T = 250		
gr_137_1	59599	45251	47381	46336.1	15.89	57663	46134	49938	48005.4	14.0	62036	47343	51008	49085.4	13.80
			T= 170					T= 160					T = 150		
gr_137_2	44238	37096	39032	38046.3	5.43	43793	36909	39785	38835.7	5.58	47417	38324	42120	39768.2	5.88
			T=365					T = 350					T= 345		
gr_137_3	64707	56243	59472	57522.1	25.25	68406	56404	61174	58269.5	24.49	59720	57852	62387	59141.7	26
			T = 235					T = 220					T = 210		
kro_150_1	19413	15429	16671	160035	19.29	21028	15761	17369	16522.5	22.17	17159	16279	18386	17042.0	20.70
			T= 155					T= 140					T = 130		
kro_150_2	11948	10597	12543	11470.0	6.88	14925	10495	14301	12091.0	6.96	13388	10989	13926	12436	6.3
			T=350					T = 340					T= 320		
kro_150_3	29057	21532	22473	21927.0	32.88	32467	21372	22923	22195	36.45	27978	22761	24747	23618.0	152.58
			T=275					T=260					T=250		
rat_195_1	1809	1317	1431	1384.0	24.36	1758	1357	1477	1411.0	23.03	1932	1346	1444	1409.0	24.57
			T= 195					T = 180					T= 170		
rat_195_2	1428	954	1046	1006.0	9.90	1207	952	1052	9967	9.21	1275	956	1066	1009	9.54
			T = 430					T=410					T = 390		
rat_195_3	2221	1854	1954	19048	61.05	2339	1872	2004	19192	55.09	2403	1891	2018	1938.0	60.68
			T= 320					T = 300					T = 290		
gr_202_1	33590	23098	24407	23796.0	51.80	37991	23284	25467	24356.0	51.01	32859	23884	28452	25277.0	57.54
			T = 210					T = 200					T = 185		
gr_202_2	26918	15680	17974	16742.0	17.46	21326	15347	17251	16411.0	16.88	24237	15347	17251	16411	16.88
			T = 455					T = 440					T = 430		
gr_202_3	46780	35308	36925	35942.0	77.22	45567	35697	37382	36349	81.38	43790	35971	38115	36786.0	58.72
			T = 405					T = 390					T = 380		
pr_264_1	66310	34828	36075	35377.0	88.45	56168	35157	36478	35575.0	77.96	48864	35345	38122	36488.0	85.19
			T = 270					T = 250					T = 240		
pr_264_2	44553	28949	30173	29498.0	31.10	44099	28839	30077	29329.0	30.05	47772	29138	30071	297357	29.6
			T = 635					T= 620					T = 610		
pr_264_3	79124	41658	43911	42789.0	188.85	75154	41656	46944	436394	187.87	68808	41553	45454	43353.0	185.64

6 Conclusion and future work

In this paper, we focus on FTSP, a new type of TSP, which is a well-known problem proposed to be adapted in real-life applications. Unlike the other works in the literature that deal with the TSP, in this study, a more real-life-like proposal is presented by defining times for the visited nodes and adding the constraint that the available time should not be exceeded while creating the route. A mathematical model of the problem is defined and a metaheuristic method is developed to approach efficient solutions in a shorter time. The proposed metaheuristic is a Variable Neighbour Descent algorithm with four different neighbourhood procedures. Two of the neighbourhood structures provide diversification and intensification by allowing small-scale changes while the other two provide diversification and intensification by allowing

large-scale changes. The tabu mechanism, on the other hand, prevents returning to the same solutions in the near future and contributes to diversification by taking the actions in the first neighbourhood to the tabu list. By applying the 2-opt algorithm on the existing solutions, the improvement rate is increased in each iteration. If no improvement is observed as the iterations progress, the perturbation operator makes significant changes to the current solution, allowing different regions to be searched. The proposed algorithm is tested using test instances from the literature. In this study, time information for each visit node was added to the existing information in the test instances. Since the mathematical model solutions require a long time, a threshold value was set for the solution time and the performance was measured according to the results found within this solution time. Each of the test cases was solved using different T values. As a result, VND achieved the best solutions in small size problems and as the problem size increased, it achieved better solutions in much shorter times compared to the mathematical model solver.

In future work, a new VND algorithm with different neighbourhood structures and more complex algorithmic improvements can be designed. Hybrid approaches can be developed to improve the performance of this algorithm. For example, simulated annealing can be used to improve initial solutions, and genetic algorithm operators can be used to provide diversity and avoid local optima. In addition, machine learning algorithms can be integrated to harmonise the algorithm with the learning processes and make it adaptive. This hybrid VND algorithm can produce more flexible and efficient solutions by adapting itself according to different problem characteristics.

For performance evaluation, making use of all datasets defined on FTSP in the existing literature allows for more comprehensive comparative analyses. In this context, the strengths and weaknesses of the algorithm can be identified by performing detailed analyses on the data sets. In addition, simulating different scenarios by creating new instances and studying the generalisability of the algorithm can increase the adaptability of the model to practical implementations.

7 Author contribution statements

In the scope of this study, Author 1 in the formation of the idea, the design, the assessment of obtained results, the literature review, supplying the data used, and examining the results.

8 Ethics committee approval and conflict of interest statement

There is no need to obtain permission from the ethics committee for the article prepared. There is no conflict of interest with any person/institution in the article prepared.

9 References

[1] Dursunoglu CF, Arslan O, Demir SM, Kara BY, Laporte G. "A unifying framework for selective routing problems". *European Journal of Operational Research*, 2024.

[2] Dantzig G, Fulkerson R, Johnson S. "Solution of a large-scale traveling-salesman problem". *Journal of the Operations Research Society of America*, 2(4), 393-410, 1954.

[3] Fischetti M, Salazar González J J, Toth P. "A branch-and-cut algorithm for the symmetric generalized traveling salesman problem". *Operations Research*, 45(3), 378-394, 1997.

[4] Golden B, Naji-Azimi Z, Raghavan S, Salari M, Toth P. "The generalized covering salesman problem". *INFORMS Journal on Computing*, 24(4), 534-553, 2012.

[5] Shaelaie MH, Salari M, Naji-Azimi Z. "The generalized covering traveling salesman problem". *Applied Soft Computing*, 24, 867-878, 2014.

[6] Pop PC, Cosma O, Sabo C, Sitar CP. "A comprehensive survey on the generalized traveling salesman problem". *European Journal of Operational Research*, 314(3), 819-835, 2024.

[7] Morán-Mirabal L F, González-Velarde JL, Resende MG. "Randomized heuristics for the family traveling salesperson problem". *International Transactions in Operational Research*, 21(1), 41-57, 2014.

[8] Papcun P, Cabadaj J, Kajati E, Romero D, Landryova L, Vascak J, Zolotova I. "Augmented reality for humans-robots interaction in dynamic slotting 'chaotic storage' smart warehouses". *In Advances in Production Management Systems*. Production Management for the Factory of the Future: IFIP WG 5.7 International Conference, APMS 2019, Austin, TX, USA, September 1–5, 2019, Proceedings, Part I, pp. 633-641, Springer International Publishing, 2019.

[9] Gonçalves JF, Resende MG. "Biased random-key genetic algorithms for combinatorial optimization". *Journal of Heuristics*, 17(5), 487-525, 2011.

[10] Festa P, Pardalos PM, Resende MG, Ribeiro CC. "Randomized heuristics for the MAX-CUT problem". *Optimization Methods and Software*, 17(6), 1033-1058, 2002.

[11] Reinelt G. "TSPLIB—a traveling salesman problem library". *ORSA Journal on Computing*, 3, 376–384, 1991.

[12] Bernardino R, Paias A. "Solving the family traveling salesman problem". *European Journal of Operational Research*, 267(2), 453-466, 2018.

[13] Bernardino R, Paias A. "Heuristic approaches for the family traveling salesman problem". *International Transactions in Operational Research*, 28(1), 262-295, 2021.

[14] Holland JH. "Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence". *MIT Press*, 1992.

[15] Bernardino R, Paias A. "The family traveling salesman problem with incompatibility constraints". *Networks*, 79(1), 47-82, 2022.

[16] Dorigo M, Birattari M, Stutzle T. "Ant colony optimization". *IEEE Computational Intelligence Magazine*, 1(4), 28-39, 2006.

[17] 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.

[18] 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.

[19] Adleman LM. "Molecular computation of solutions to combinatorial problems". *Science*, 266(5187), 1021-1024, 1994.

[20] Lipton RJ. “DNA solution of hard computational problems”. *Science*, 268(5210), 542-545, 1995.
[21] Nourmohammadzadeh A, Sarhani M, Voß S. “A matheuristic approach for the family traveling salesman problem”. *Journal of Heuristics*, 29(4), 435-460, 2023.
[22] Kirkpatrick S, Gelatt Jr CD, Vecchi MP. “Optimization by simulated annealing”. *Science*, 220(4598), 671-680, 1983.
[23] Ribeiro CC, Hansen P, Taillard ED, Voss S. “POPMUSIC-Partial optimization metaheuristic under special intensification conditions”. *Essays and Surveys in Metaheuristics*, 613-629, 2002.
[24] Domínguez-Casasola S, González-Velarde JL, Ríos-Solís YÁ, Reyes-Vega KA. “The capacitated family traveling salesperson problem”. *International Transactions in Operational Research*, 31(4), 2123-2153, 2024.
[25] Chaves AA, Vianna BL, da Silva TT, Schenekemberg CM. “A parallel branch-and-cut and an adaptive metaheuristic to solve the Family Traveling Salesman Problem”. *Expert Systems with Applications*, 238, 121735, 2024.
[26] Chaves AA, Lorena LHN. "An adaptive and near parameter-free BRKGA using Q-learning method". *2021 IEEE Congress on Evolutionary Computation (CEC)*, Kraków, Poland, 2021, June 28.
[27] Pandiri V, Singh A. “Solution of the family traveling salesman problem using a hyper-heuristic approach”. *Engineering Applications of Artificial Intelligence*, 133, 108193, 2024.
[28] Shaw P. “Using constraint programming and local search methods to solve vehicle routing problems”. *In International Conference on Principles and Practice of Constraint Programming*, pp. 417-431, Berlin, Heidelberg, 1998.
[29] Miller CE, Tucker AW, Zemlin RA. “Integer programming formulation of traveling salesman problems”. *Journal of the ACM*, 7(4), 326-329, 1960.
[30] Şahin Y, Karagül K. “Gezgin satıcı probleminin melez akışkan genetik algoritma (MAGA) kullanarak çözümü”. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 25(1), 106-114, 2019.
[31] Wen X, Zhang X, Xing H, Ye G, Li H, Zhang Y, Wang H. “An improved genetic algorithm based on reinforcement learning for aircraft assembly scheduling problem”. *Computers & Industrial Engineering*, 110263, 2024.
[32] Hansen P, Mladenović N. “An introduction to variable neighborhood search”. *Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pp. 433-458. Boston, MA: Springer US, 1999.
[33] Crispim J, Brandão J. “Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls”. *Journal of the Operational Research Society*, 56(11), 1296-1302, 2005.
[34] Glover F. “Tabu search—part I”. *ORSA Journal on Computing*, 1(3), 190-206, 1989.