Exploring Set Partitioning in Combinatorial Optimisation: Revisiting a TSP Example

H.A. Krüger[1*] & I. Mayer[1]

ARTICLE INFO

Article details
Submitted by authors 6 May 2025
Accepted for publication 13 Apr 2026
Available online 22 May 2026

Contact details
* Corresponding author
Hennie.Kruger@nwu.ac.za

Author affiliations
1 School of Computer Science and Information Systems, North-West University, Potchefstroom, South Africa

ORCID® identifiers
H.A. Krüger
https://orcid.org/0000-0001-8514-4422

I. Mayer
https://orcid.org/0009-0002-8178-5403

DOI
http://dx.doi.org//10.7166/37-1-3242

ABSTRACT

Large-scale combinatorial problems are often computationally intractable, and alternative solution approaches are frequently used to find effective approximate solutions for these problems. This paper introduces a six-step methodology, adapted from an existing framework in the literature, and based on a set partitioning scheme, aimed at generating near-optimal solutions for combinatorial problems. The proposed approach specifically considers the classic travelling salesman problem, and the methodology's performance is demonstrated on a real-world 2508-node instance. The results confirm that the proposed set partitioning approach yields satisfactory solution quality and offers practical implementation advantages.

OPSOMMING

Grootskaalse kombinatoriese probleme is dikwels rekenaarmatig onhanteerbaar, en alternatiewe oplossings-benaderings word gereeld gebruik om doeltreffende benaderde oplossings vir hierdie probleme te vind. Hierdie studie stel 'n ses-stap metodologie voor wat aangepas is vanuit 'n bestaande raamwerk in die literatuur. Die metode is gebaseer op 'n partisieskema wat daarop gemik is om byna-optimale oplossings vir kombinatoriese probleme te genereer. Die voorgestelde benadering fokus spesifiek op die klassieke reisende handelsreisiger probleem en die metodologie se prestasie word gedemonstreer op 'n werklike 2508-node geval. Die resultate bevestig dat die voorgestelde partisieskema bevredigende oplossingsgehalte lewer en praktiese implementerings-voordele bied.

1. INTRODUCTION

Combinatorial optimisation is a branch of mathematics, and the term refers to a well-known class of optimisation problems in which the aim is to find an optimal solution from a large discrete (finite) set of possible solutions [1]. The problem is characterised by a sequential decision-making process in which a series of discrete choices are made to maximise or minimise a pre-specified objective function subject to a set of constraints. Mathematically, the problem may be expressed as follows:

Consider a finite set S of possible solutions, and an objective function $f: S \to \mathbb{R}$. The goal is to find an element $x^* \in S$ such that:

$$x^* = argmax(or\ argmin)_{x \in S} f(x),$$

depending on whether the problem is a maximisation or minimisation problem. In this formulation, S represents a finite discrete search space, $f(x)$ the objective function associated with solution x, and x^* the final optimal choice.

Many combinatorial problems are classified as non-deterministic polynomial-time-hard (NP-hard) – i.e., they cannot solve all instances in polynomial time [2]. NP-hard problems are computationally intractable for large problem instances owing to the exponential growth of the solution space. For example, the

solution space in a travelling salesman problem, with n cities, is $(n-1)!/2$, and it grows factorially with n. Exact algorithms are therefore often impractical for large-scale problems, and techniques such as polyhedral methods, branch-and-bound, and cutting plane algorithms can solve only relatively small problem instances to optimality. In practical applications, the computational difficulties of exact algorithms have led to the development of approximation algorithms, heuristics, and metaheuristics that offer near-optimal solutions while significantly reducing the computational effort. Methods such as genetic algorithms [3], simulated annealing [4], and others are widely used as solution alternatives for large-scale combinatorial optimisation problems with the goal of balancing computational efficiency and solution quality.

Despite the computational complexities, combinatorial optimisation remains an important area of study due to the pivotal role it plays in various real-world applications. The development and improvement of numerous approximation algorithms and heuristics is an active area of research [1], while practical applications are widespread in areas such as telecommunications [5], logistics and supply chain management [6], manufacturing [7], vehicle routing [8], energy management [9], scheduling [10], biology [11], and robotics [12]. Advances such as parallel computing, quantum computing, and the development of specialised hardware [13] have expanded the scope of combinatorial optimisation applications and solution methods. These advancements ensure that combinatorial types of discrete decisions, where resources are limited, can be solved in an efficient and cost-effective manner.

Numerous approaches and methodologies have been developed to address computationally expensive combinatorial problems. In this paper, the focus is on a set partitioning approach, which is a form of integer optimisation problem [14]. The technique involves dividing a given set into several subsets that are both mutually exclusive and collectively exhaustive. This ensures that no element of the original set is included in more than one subset, and that the union of all subsets contains every element of the original set. The set partitioning method, which is formally defined and described in Section 3, has been applied successfully in many application areas such as scheduling [15], resource allocation [16], and clustering problems [17]. The objective of this paper is to reconfirm the applicability of a set partitioning approach in finding an approximate solution to an NP-hard combinatorial optimisation problem by examining an illustrative example. Reconfirming the set partitioning approach would help to validate the method's assumptions, ensure the generalisability of the method, and help to understand its possible limitations in different application areas. Specifically, the study focuses on the widely studied travelling salesman problem (TSP), and the set partitioning method used is adapted from the approach suggested by Soroudi [18] to improve and enhance its applicability to real-world scenarios while reducing computational overhead. The proposed method is then applied to a real-world dataset. Through the real-world example, the paper seeks to demonstrate and reaffirm the practical effectiveness and applicability of set partitioning approaches in producing 'good enough' solutions to combinatorial optimisation tasks.

The remainder of the paper is structured as follows. Section 2 presents a brief literature contextualisation, while Section 3 defines and explains the concept of set partitioning. Section 4 is the focus of the paper, and introduces the TSP as the route optimisation application, the proposed set partitioning approach, and the real-world case study. A discussion of the results is given in Section 5, and the paper ends with some final remarks in Section 6.

2. LITERATURE REVIEW

The purpose of this section is to contextualise the contents of the paper by referencing only a few of the studies related to the main topics of the paper. These topics are combinatorial optimisation, set partitioning, and the TSP.

2.1. Combinatorial optimisation

Combinatorial optimisation is the overarching concept in this paper. The significance of the approach, particularly the use of heuristics and special techniques to produce approximate rather than exact solutions, has received considerable attention from researchers. Examples of the many application areas in which combinatorial optimisation is the focus have already been mentioned in the introduction. In addition to those, the work of Kaya *et al*. [19] refers to at least twelve principal combinatorial optimisation application areas. These areas range from assembly problems, bioinformatic problems, and graph colouring problems to routing and social network analysis applications. Special mention is also made of the TSP as a combinatorial optimisation problem. To gain a good understanding of the combinatorial optimisation concept, the work of Mazyavkina *et al*. [2] and of Sanchez *et al*. [20] may be consulted. The former study

describes the formulation of different combinatorial optimisation problems (such as the TSP) and the potential use of reinforcement learning algorithms to solve these types of problem; while the latter study offers explanations of specific combinatorial problems such as bin packing and job shop scheduling, together with a review of hyper-heuristics as a solution technique. The prominent role that heuristic and metaheuristic solutions play in approximating combinatorial solutions is noticeable in the large number of recorded studies in the literature. Examples include nature-inspired metaheuristics; see, for example, Rahman *et al*. [21], who provide a systematic review of these types of metaheuristics. Other popular metaheuristics are evolutionary and genetic algorithms [22], [23] and simulated annealing [4].

2.2. Set partitioning

Set partitioning is a well-known approach that is used by many researchers to address various combinatorial optimisation problems. As far back as 1972, Marsten [14] documented a long list of application domains in which set partitioning approaches were used. Examples of these areas range from scheduling problems, switching theory and stock cutting to capital investment and facility location problems. Set partitioning is defined in the next section; the objective here is to present only a few examples of the literature's resources to provide some background to the relevance and importance of set partitioning, particularly in the context of combinatorial optimisation, as a solution methodology.

Some examples of how set partitioning is used in different application areas are the following. Mingozzi *et al*. [24] leveraged set partitioning to address crew scheduling problems, while Cattrysse *et al*. [25] applied the technique to solve the general assignment problem. Matatov *et al*. [26] explored the use of set partitioning in data mining, and Grenouilleau *et al*. [15] used it to resolve home care and scheduling problems. Furthermore, Borisovsky *et al*. [27] demonstrated the usefulness of set partitioning in balancing the control of machine lines. Beyond its application to specific problem areas, a significant number of research studies have been dedicated to enhancing and refining set partitioning methodologies; see, for example, the studies by Michalak *et al*. [28] and Lewis *et al*. [29]. Readers seeking more information on some of the theoretical underpinnings and algorithmic solution techniques (both heuristic and exact approaches) of set partitioning are referred to the works in [30] and [31].

2.3. Travelling salesman problem

The TSP is one of the most extensively studied combinatorial optimisation problems. Because of its practical and real-world significance, the TSP was selected for this study as a suitable example to demonstrate the proposed set partitioning approach. While a more detailed discussion of the TSP is provided in Section 4.1, this brief contextual overview highlights only its flexibility and the various problem variants that have already been introduced in the literature on the TSP.

The *asymmetric* TSP is a variant in which the bidirectional distances between pairs of nodes (visiting points) are not necessarily the same, meaning that the travelling cost may differ when travelling in the opposite direction [32]. Another widely studied variant that allows for flexible routing strategies is the *generalised* TSP; in this problem scenario, not all nodes are required to be visited [33]. In certain instances, more than one salesman is used to visit different subsets of nodes; in these cases, the problem is formulated as a *multiple* TSP [34]. When there is a need to visit a specific node several times over a given time horizon, the TSP model is referred to as a *period* TSP [35]; and, in logistics and transportation contexts, when a vehicle may both collect and deliver items or products to the nodes, the problem becomes a *pickup-and-delivery* TSP [36]. Traditional TSP models assume that all nodes must be visited exactly once; however, whenever a node has a predetermined probability (less than 1) of requiring a visit, a *probabilistic* TSP model is used [37]. The existence of a visiting order (typically because of delivery or dependency requirements) to nodes requires a TSP with *precedence constraints* [38], while a TSP with *time windows* is used in instances when visiting a node is subject to both a latest arrival time and an earliest arrival time [39].

In addition to the many application areas, the TSP is also widely studied to improve and refine existing solution methods and to develop new solution techniques. Solution methods include both exact and heuristic approaches, and much focus is also placed on approximation methods to address large-scale TSP problem instances [40]. The work of Laporte [41] provides a good overview of both exact and heuristic solutions to the TSP.

3. SET PARTITIONING

A set partitioning problem involves dividing a given set into non-overlapping subsets such that every element of the original set belongs to exactly one subset. Mathematically, this can be expressed as follows [42]:

> Consider the index set $\mathcal{I} = \{1, 2, \dots, m\}$. Let $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$ denote a set of n subsets and let $\mathcal{P} \subset \{1, 2, \dots, n\}$. Then, $\mathcal{P}$ defines a partition of $\mathcal{I}$ if and only if $\bigcup_{j \in \mathcal{P}} S_j = \mathcal{I}$ and $S_j \cap S_k = \emptyset, \forall j, k \in \mathcal{P}, j \neq k$.

The set partitioning concept can be illustrated by representing the problem as a matrix. Each vector (column) of a matrix A represents a subset S_j. The size of each vector is equal to m and the vector contains only 0s and 1s. Element i in a vector is 1 if $i \in S_j$ and 0 otherwise. Figure 1 shows a hypothetical matrix representation of the set partitioning problem.

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	.....	c_n		
1	1	0	1	1	0	0	1	0	.....	0	=	1
2	0	1	0	0	0	1	0	0	.....	0	=	1
3	0	0	1	0	1	0	0	0	.....	0	=	1
.												.
.												.
.												.
m	0	0	0	0	0	1	1	0	.....	1	=	1

Figure 1: Hypothetical matrix representation

The set partitioning problem is formulated as an integer linear programming model as follows [14]:

$$\text{Minimise} \quad z = \sum_{j=1}^{n} c_j y_j \tag{1}$$

subject to

$$\sum_{j=1}^{n} a_{ij} y_j = 1 \qquad i = 1,2, \dots, m \tag{2}$$

$$y_j \in \{0, 1\} \qquad j = 1,2, \dots, n. \tag{3}$$

An element a_{ij} = 1 if column j covers row i and 0 otherwise (see also Figure 1). The cost associated with vector S_j is indicated by c_j and the goal of the model is then to select a set of vectors (S_j), which will result in a minimal cost partition of the index set $\mathcal{I} = \{1, 2, \dots, m\}$. It should be noted that constraint set (2) is frequently replaced by $\sum_{j=1}^{n} a_{ij} y_j \geq 1$. This results in the traditional *set covering* problem. If a '≤' inequality sign is used, the problem is referred to as a *set packing* problem.

A set partitioning problem is a well-known NP-hard problem, as the number of possible partitions grows exponentially when the original set size increases. This makes it computationally expensive to solve, and approximate solutions, based on heuristics, are often preferred to exact algorithms. The TSP example in the sections that follow is based on the set partitioning principles presented in this section.

4. THE TSP AS ROUTE OPTIMISATION APPLICATION

This study adopts the TSP as the route optimisation application to demonstrate the proposed set partitioning approach.

There are several advantages to using the classic TSP to illustrate a new proposed approach. First, its real-world relevance is evident, as numerous practical and real-world scenarios align directly with the TSP framework, such as delivery routes, logistics, and circuit board design. Second, its computational complexity (arising from its NP-hard nature) makes the TSP an ideal testbed for evaluating novel or adapted solution methodologies. Third, the versatility of the TSP facilitates experimentation with diverse solution strategies, ranging from locally optimal approaches such as greedy algorithms to more sophisticated

metaheuristic frameworks such as simulated annealing and genetic algorithms. In addition, the TSP's status as a well-researched and widely studied problem brings several secondary benefits. For instance, using the TSP as an example allows for the performance of newly proposed methodologies to be benchmarked against established standards. It also enables a comprehensive analysis of a new method's behaviour in a variety of problem instances. Moreover, developing new approaches to a problem such as the TSP contributes to the broader field of route optimisation, and may inspire further research.

This section forms the focus of the proposed set partitioning approach as applied to the TSP domain. It opens with a brief description of the classic TSP formulation in the form of an integer linear programming model (Section 4.1). This is followed by a description of the proposed methodology (Section 4.2), and the section concludes with a real-world illustration of the proposed process (Section 4.3).

4.1. The traditional TSP

Conceptually, the traditional TSP can be framed as finding the shortest possible route for a salesperson to visit n cities and return to the point of origin. A more formal definition is as follows:

> Consider a graph $G = (V, A)$, where V denotes a set of n vertices and A represents the corresponding set of edges or arcs. Associated with A is a distance matrix $C = (c_{ij})$, which specifies the distances (costs) between all pairs of vertices. The TSP involves identifying a minimum-distance (cost) circuit that traverses each vertex exactly once before returning to the initial vertex. This circuit, commonly referred to as a 'tour' or 'Hamiltonian cycle', represents the optimal path. The problem is normally described as a symmetrical problem if $c_{ij} = c_{ji}$ for all $i, j \in V$.

The classic TSP has been represented using two prominent integer linear model formulations. The first model, known as the DFJ formulation, was introduced by Dantzig, Fulkerson and Johnson in 1954 [43]. Six years later, Miller, Tucker and Zemlin proposed an alternative model, now referred to as the MTZ formulation [44]. Both formulations serve as standard integer programming models for the TSP, differing primarily in the structure of the subtour elimination constraints. In this study, the MTZ formulation is presented and briefly discussed.

Consider an n-city TSP scenario, and define the following variables:

$$x_{ij} = \begin{cases} 1 & if\ egde\ (i,j)\ is\ included\ in\ the\ TSP\ path \\ 0 & otherwise \end{cases}$$

$u_i =$ a set of variables representing arbitrarily non $-$ negative numbers.

The standard MTZ TSP model is then formulated as follows.

$$\text{Minimise} \quad z = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} c_{ij} x_{ij} \tag{4}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, 2, \ldots., n; i \neq j, \tag{5}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, 2, \ldots., n; i \neq j, \tag{6}$$

$$u_i - u_j + n x_{ij} \leq n - 1 \qquad i, j = 1, 2, \ldots, n;\ i \neq j, \tag{7}$$

$$x_{ij} \in \{0,1\} \qquad i, j = 1, 2, \ldots, n, \tag{8}$$

$$u_i \geq 0 \qquad i = 1, 2, \ldots, n. \tag{9}$$

The objective function, given in equation (4), minimises the total distance travelled in the TSP route. Constraint set (5) requires that the salesman arrives at each city exactly once before returning to the origin. Similarly, constraint set (6) guarantees that the salesman departs from each city exactly once. To eliminate subtours, constraint set (7) introduces auxiliary variables and enforces the necessary conditions to maintain

a single connected tour. The binary nature of the decision variables x_{ij}, representing whether the edge between cities i and j is included in the route, is enforced by constraint set (8). Finally, constraint set (9) defines the non-negative auxiliary variables used in the subtour elimination constraints.

The number of variables and the number of constraints in the standard TSP model formulated in (4) – (9), serve as an indicator of the computational complexity inherent in solving the problem. This complexity underscores the importance of exploring alternative solution methodologies, such as set partitioning approaches, which can generate satisfactory approximate solutions more efficiently. For instance, in the standard MTZ n-city model formulation of the TSP above, the model includes $n^2 - 1$ variables. The number of constraints depends on the symmetry of the problem. In the case of a symmetric TSP, where $distance(i,j) = distance(j,i)$, the model contains $[n(n+1)]/2$ constraints. This significant constraint count emphasises the computational problem posed by the standard TSP formulation.

The original paper and a more detailed discussion of the MTZ model formulation can be found in [44].

4.2. The proposed set partitioning process

This section provides a detailed description of the proposed set partitioning process, which consists of six primary steps. The methodology draws on the framework suggested by Soroudi [18], with significant adaptations in each step to enhance its robustness and computational efficiency. The modifications were introduced to align the method with real-world requirements, and aimed to make the process not only more effective but also more suitable for practical implementation. In Section 5, a comparison of the steps in the proposed method and the steps in the original approach outlined in [18] is presented.

The six steps making up the set partitioning methodology are as follows:

- Identify and obtain a suitable dataset of locations;
- Partition the dataset into subsets;
- Determine the sequence of partition evaluations;
- Establish the optimal connections between successive partitions;
- Evaluate/solve each partition; and
- Integrate partition evaluations to generate an approximate overall solution.

Following the detailed description of each step, a real-world case study involving a dataset of 2 508 locations is introduced to illustrate the method's capacity to reduce computational complexity by generating an approximate solution.

4.2.1. Step 1: Identify and obtain suitable dataset of locations

To demonstrate the proposed set partitioning approach, both synthetic and real-world datasets are used. For the real-world application, a dataset containing the coordinates of actual locations is used, as detailed in Section 4.3.1. However, to illustrate and explain the methodology, a synthetic dataset comprising 2 000 points is generated within a 200 × 200 Cartesian plane. Two scenarios are considered: first, a uniform distribution, in which the generated points are evenly distributed across the plane; and second, a skewed distribution, in which points are concentrated in specific areas to simulate higher density. The latter scenario is more representative of real-world conditions, where locations are seldom evenly distributed around a region. Figure 2 presents a visual representation of these two datasets, which form the basis for explaining the remainder of the steps in the proposed approach.

Once a suitable dataset has been identified (either a synthetic or a real-world dataset), the proposed process proceeds with the partitioning of the dataset into subsets. This partitioning step is detailed in the next sub-section.

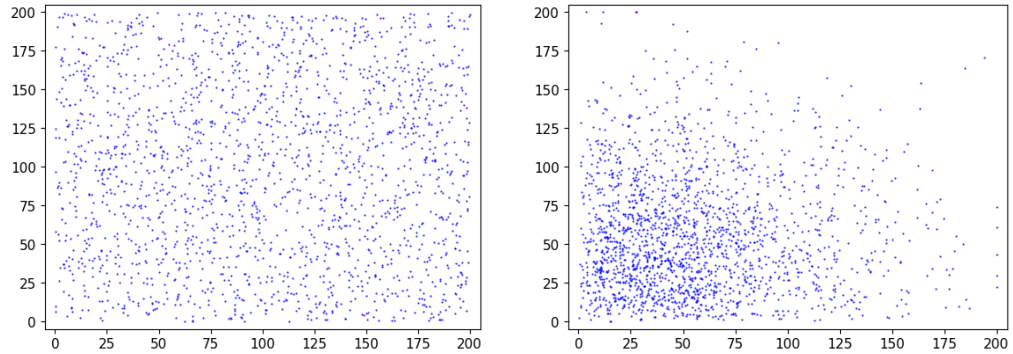

Figure 2: Hypothetical 2 000 locations dataset for illustration purposes. A uniformly distributed dataset is shown on the left, and a more realistic skewed distribution on the right.

4.2.2. Step 2: Partition the dataset into subsets

As outlined in the introduction, partitioning a dataset involves dividing it into subsets that are mutually exclusive and collectively exhaustive. This ensures that each element of the original dataset is assigned to exactly one subset, while the union of all subsets encompasses the entire dataset.

Several methods exist to partition a set of points while adhering to these principles. A straightforward approach is to superimpose a grid of uniform, equally sized squares on the 200 × 200 Cartesian plane containing the points [18]. Alternative methods include adaptive partitioning techniques based on point density or clustering-based approaches. In this paper, a quadtree partitioning scheme was used to create a grid of cells over the Cartesian plane. Unlike uniform grids, the quadtree method generates partitions of varying sizes, reflecting the spatial distribution of points. This characteristic makes it particularly effective for non-uniformly distributed datasets, as it dynamically adjusts the number of partitions, and therefore also the partition sizes according to point density. Consequently, quadtree partitioning is widely regarded as an efficient algorithm for tasks such as spatial indexing and searching tasks, especially in real-world applications where uniform point distributions are rare. The versatility of quadtree partitioning is evidenced by its applications in fields such as geographic information systems [45], image compression [46], and computer graphics [47].

The quadtree algorithm operates by recursively subdividing a two-dimensional space into four equal-sized quadrants, forming a hierarchical tree structure. This process continues until a stopping criterion, such as a minimum number of points per cell, is met. Applying the quadtree partitioning technique to the two illustrative datasets yielded the partitioning results shown in Figure 3.

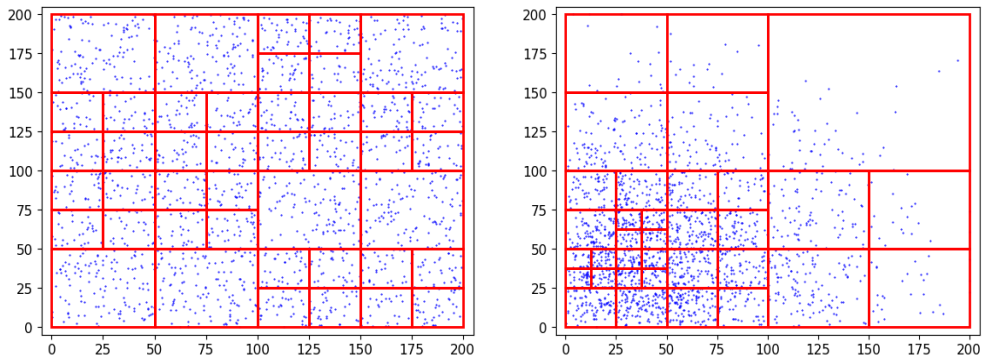

Figure 3: Quadtree partitioning. The left-hand figure shows the partitioning for the uniformly distributed dataset, and the one on the right is the partitioning for the skewed distribution.

The advantages of the approach are evident on the right in Figure 3, where the partitioning (34 partitions) adapts to the density of points. For instance, in the densely populated lower-left corner, the number of

partitions (grid cells) is significantly higher than the sparsely populated upper-right corner. The grid on the left (the uniform distribution) was artificially manipulated to produce 43 partitions for illustration purposes, as the quadtree partitioning would have produced simply a set of even-sized grid cells owing to the perfect uniformly distribution of the datapoints.

4.2.3. Step 3: Determine the sequence of partition evaluations

Partitioning the problem requires a logical sequence for evaluating the partitions, particularly during the final stages of the process when individual partition solutions are integrated into a complete solution. The integration of partitions cannot occur arbitrarily, but should follow a sequential order that contributes to the overarching objective of optimising the solution. It is therefore essential to determine an appropriate sequence for partition evaluation that aligns with the final objective of finding the best route.

To establish a meaningful sequence for partition evaluation, the midpoints of each partition are computed. These midpoints, representing the centre of each grid cell, are treated as nodes in a standard TSP. For the illustrative skewed distribution considered here, 34 midpoints are calculated, corresponding to the 34 grid cells. The sequence of partition evaluations is then determined by solving a TSP, with the partition at the origin (corner point (0,0)) designated as both the starting and the ending node. The solution to this TSP provides the optimal sequence of partition evaluations, which in turn facilitates the integration and connection of individual partition solutions into an overall solution.

The standard binary integer programming model presented in (4) – (9) for the TSP can be used to solve this problem; it would require 1 155 variables for the 34-node case. However, to mitigate computational complexity, a heuristic approach is preferred. Most conventional heuristics, such as the greedy algorithm, yield routes with crossing paths, which would be unsuitable, as a circular route without crossings is required. To address this, a secondary optimisation heuristic, such as the 2-opt algorithm [48], can be applied to eliminate path crossings.

Given the availability of the partitions and their neighbouring relationships, a depth-first search (DFS) strategy was selected to generate the required route. DFS is a recursive algorithm that explores the vertices of a graph or tree structure based on adjacency relationships, ensuring that no path crossings occur [49]. Although DFS does not guarantee the shortest path, its performance is generally effective for small-scale problems such as the one under consideration. To validate the results, the problem was also solved using the Christofides algorithm, a well-established approximation algorithm for the TSP [50], [51]. The empirical results showed that the DFS approach could improve on the Christofides solution by up to 20% in this specific case. It is important to note that the choice of algorithm for solving the midpoint TSP is flexible, and that this study does not claim that either the DFS or the Christofides algorithm represents the optimal approach. Rather, any suitable heuristic or model could be used to achieve a circular route.

Figure 4 illustrates the optimal circular routes obtained for the midpoint TSP under both uniform and skewed distributions of points.

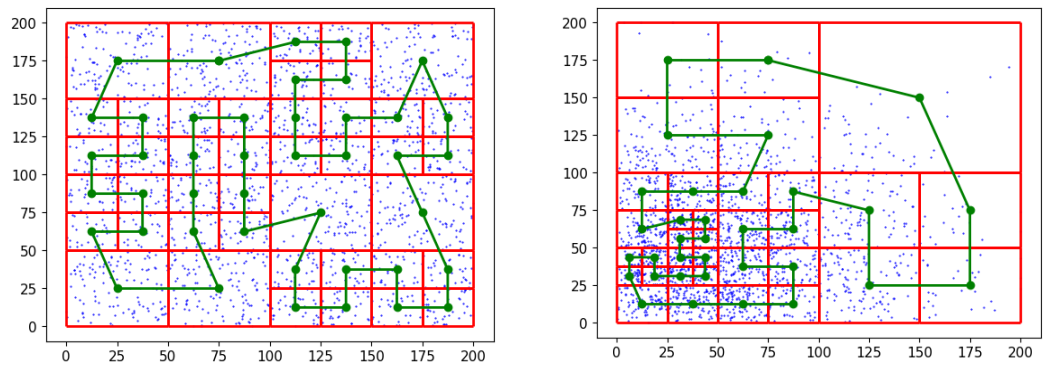

Figure 4: Sequence of partition evaluations. On the left is the sequence for the uniformly distributed dataset, and on the right is the sequence for the skewed distribution.

The sequence for evaluating the partitions in Figure 4, for both distributions, would start at the lower left partition, and could then proceed to any of the connecting neighbours. Because it is a circular route, it does not matter which one of the two connecting neighbours is selected.

Before evaluating the partitions in accordance with the sequence specified by the TSP solution, it is necessary to identify the optimal approach to connecting these partitions in the final solution. Specifically, this involves determining which points in successive partitions should be used to establish connections that contribute to achieving the objective of minimising the overall path length. The determination of these optimal connections between consecutive partitions is explained in the next step.

4.2.4. Step 4: Establish the optimal connections between successive partitions

Once the sequence of partition evaluation has been established in Step 3, the optimal connections between successive partitions must be determined. For example, if partition P_1 is followed by partition P_2, then a point (node) in partition P_1 must be determined that can be linked to a point in partition P_2 in such a way that the link between the two points represents the 'best' link, measured against the distances between all points in partitions P_1 and P_2. The obvious way to do this is to perform a pairwise comparison of the distances between all points in P_1 and all points in P_2 and simply select the two points in the two partitions with the shortest distance as the connection between the two partitions.

To save on computational time, a simple technique that evaluates only a few points, as opposed to a pairwise comparison of the distances between all points in the two partitions, was implemented. The points evaluated were those located closest to the partitioning border, and the objective was to find the shortest distance between only these border points. Once the two points closest to each other had been identified, it was unnecessary to perform further pairwise comparisons between the remaining data points. This could be explained as follows:

Suppose that the closest two points to the partitioning border are x_1 (in partition P_1) and y_1 (in partition P_2). The distance between these two points is known and is represented by $d(x_1, y_1) = d_1$. For any other point x_2 in partition P_1, for which $d(x_1, x_2) = d_2$ is greater or equal than $d(x_1, y_1)$, $i.e.$, $d_2 \geq d_1$, the following is valid: $d(x_2, y_1) = d_3 \geq d_1$; this implies that the point x_2 does not need to be considered as a connection point in partition P_1. See Figure 5 for a graphical explanation.

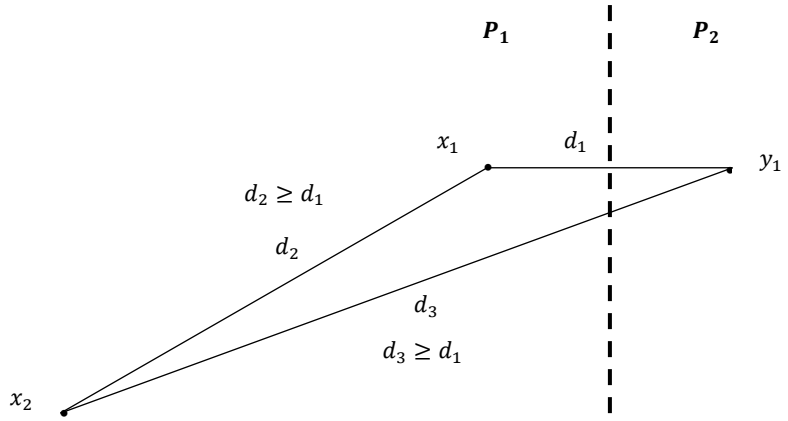

Figure 5: Distance-based connections between partitions

The proof of this assumption is intuitive. If $d(x_1, x_2) \geq d(x_1, y_1)$, then x_2 cannot lie between x_1 and y_1 or on the line segment connecting them. Furthermore, geometrically it can be observed that $d(x_2, y_1) \geq d(x_1, y_1)$ to ensure that a closed triangle shape is formed that satisfies the standard triangle inequality requirements. Using this technique, significantly reduces the number of pairwise comparisons when determining the best connection points between two partitions. The optimal connection points between partitions obtained by applying the proposed technique are shown in Figure 6.

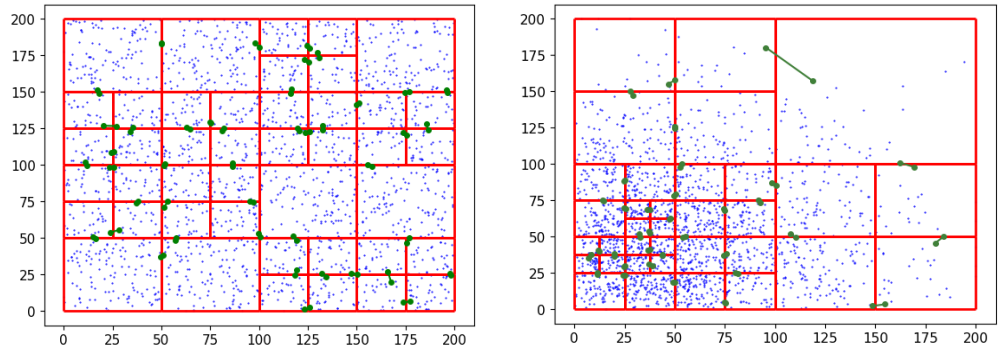

Figure 6: Optimal connections between partitions

The green nodes in Figure 6 denote the optimal connection points between successive partitions. On the left is the optimal connections for the uniform distribution and on the right for the skewed distribution.

4.2.5. Step 5: Evaluate/solve each partition

The penultimate stage of the proposed methodology involves determining the optimal route in each partition, using the identified connection points in the previous step as start and end points for the tour. While a depth-first search approach was used previously (Step 3) for inter-partition pathfinding, its application at this partition level was deemed computationally prohibitive. The potential number of data points in individual partitions would necessitate the construction of a graph for each partition; and, coupled with the maintenance of neighbour adjacency information for each point, excessive computational overhead and complexity would be created. Consequently, the computationally less-demanding nearest-neighbour heuristic was adopted to generate an initial route in each partition. This initial path was then optimised by using a 2-opt algorithm [48]. The 2-opt algorithm offers two key advantages. First, it significantly improves the initial path length (experimental results demonstrating improvements of up to 22%). Second, it effectively eliminates route crossings, a potential result of the nearest-neighbour heuristic. The optimised path for each partition obtained is presented in Step 6.

4.2.6. Step 6: Integrate partition evaluations to generate an approximate overall solution

The final step of the proposed methodology involves integrating the partition evaluations obtained in the preceding step. Specifically, all partitioned routes are now connected using the optimal connection points identified earlier in Step 4, forming a cohesive overall solution. The resulting solution constitutes a circular route for the initial set of points. In the context of the TSP, any point in this route can serve as the starting and ending location. The final overall solutions are presented in Figure 7.

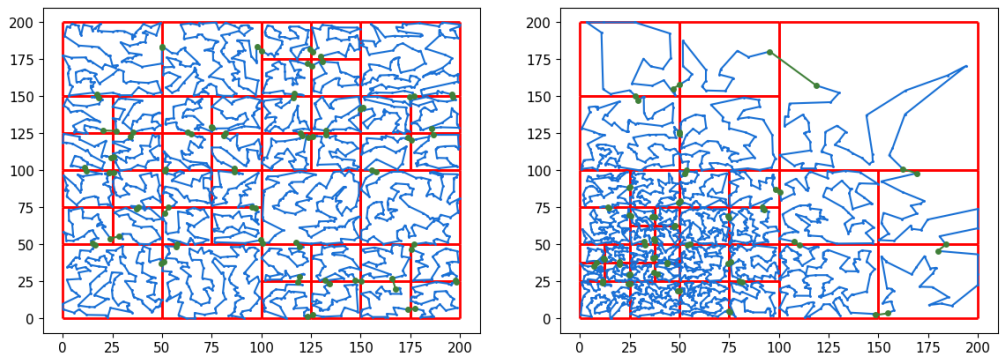

Figure 7: Final approximate solution. On the left is the final approximate solution for the uniformly distributed dataset, and on the right is the final approximate solution for the skewed distribution.

It is important to emphasise that the obtained solution is an approximation, designed to address the computational complexity associated with large-scale instances. The proposed approach is particularly beneficial when the number of nodes is substantial and when exact solutions may become computationally infeasible. The heuristic techniques used in some of the steps, such as the greedy nearest-neighbour algorithm and the 2-opt algorithm, could be replaced with alternative algorithms, based on the modeller's preference.

4.3. Real-world application

The effectiveness of any newly proposed approach or methodology to solving a specific problem should normally be assessed using a real-world case study. Empirical validation with a practical problem instance is important to ensure that the proposed technique can handle the complexities inherent in real-world scenarios, which often differ significantly from theoretical models. A real-world application not only demonstrates the practical relevance, usability, and feasibility of the method, but also highlights its potential for addressing real-world problems. Furthermore, evaluating the methodology on a real-world dataset could lead to new research opportunities and establish a benchmark for future studies in the field.

This section presents a real-world application of the proposed six-step set partitioning process. A description of the real-world dataset that was used, along with the results obtained, is provided in the two sub-sections below.

4.3.1. The dataset

The dataset used in this case study was derived from publicly available store location data of Pep Africa, the largest single-brand retailer in Africa. The case study specifically focused on store locations in South Africa, Namibia, Botswana, Eswatini, and Lesotho, resulting in a total sample size of 2 508 usable store locations. The retailer also operates in other Southern African countries such as Angola, Malawi, Mozambique, and Zambia, but these locations were not considered in the case study.

The store location data was acquired from the Pep Stores official website through a combination of Google Maps functionality and the store locator information available on the retailer's webpage. Each store's geographical position was recorded as latitude and longitude coordinates. To facilitate computational processing, these coordinates were transformed into numerical values and scaled to fit a 200 x 200 Cartesian plane. This transformation enabled an effective evaluation and presentation of the proposed partitioning methodology.

The sample size of 2 508 store locations was considered sufficiently large to illustrate effectively the proposed partitioning approach. Notably, in the context of a TSP scenario, incorporating this number of locations would result in more than 6.29 million variables and over 3.146 million constraints for a symmetric TSP instance under the Miller-Tucker-Zemlin model formulation presented in Section 4.1. These figures underline the computational complexity of the problem, and provide a rationale for adopting an approximation method such as the one proposed in this study.

A graphical representation of the store locations is presented in Figure 8. Owing to the high density of store locations, particularly in larger metropolitan areas, certain stores may not be individually distinguishable in the plot, as overlapping store plots may obscure some locations.

The location dataset shown in Figure 8 represents the first step of the proposed set partitioning approach (i.e., to obtain a suitable dataset), and is used in the next section to demonstrate the practical relevance and useability of the remaining steps of the methodology.

4.3.2. Application and results

The initial phase of the proposed methodology involved acquiring a suitable dataset, which was addressed in the preceding section, with the dataset visually represented in Figure 8. The next four stages of the partitioning approach encompassed the partitioning of the dataset; the determination of the sequence in which partitions were evaluated; the identification of optimal connection points between partitions; and the evaluation of each partition. These steps were executed as detailed in Sections 4.2.2 – 4.2.5, and the specific implementation of each step is not repeated in this section. The final stage of the approach involves integrating the partition evaluations to generate an approximate solution, and is illustrated in Figure 9.

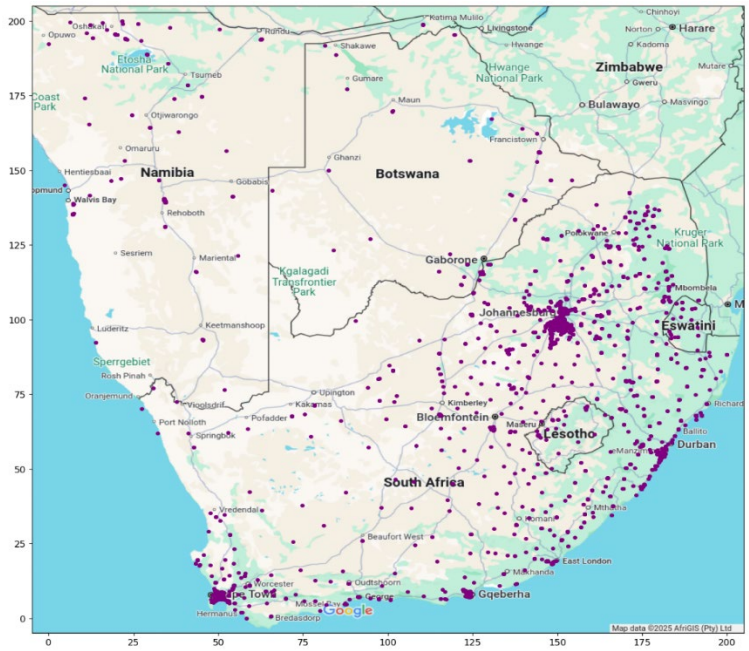

Figure 8: Store locations used in the case study

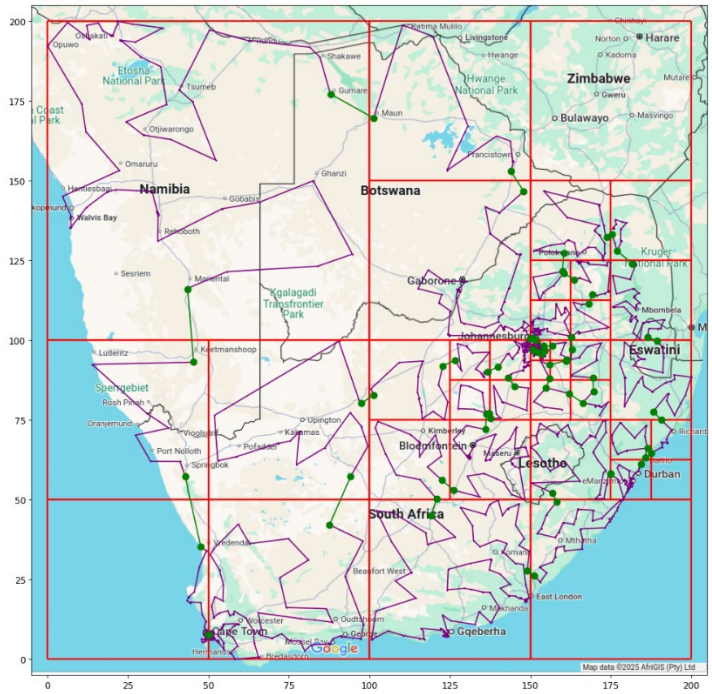

Figure 9: Final approximation solution obtained by the proposed set partitioning approach

A total of 35 partitions was used, and the resulting approximated tour length, depicted in Figure 9, is 3 105.36 units in the context of the 200 × 200 Cartesian plane. The computed route forms a closed-loop structure, allowing any location point in a partition to be designated as both the starting and the ending node for the TSP.

The successful implementation of the proposed set partitioning methodology in this real-world TSP case study highlights not only the effectiveness and useability of the technique, but also strengthens the broader

applicability of set partitioning approaches in generating approximate solutions to combinatorial optimisation problems.

5. DISCUSSION OF RESULTS

The results of the practical case study are self-explanatory, and provide clear insights into the effectiveness of the proposed approach. This section focuses on how the original framework introduced by Soroudi [18] has been adapted and improved to enhance its applicability to real-world scenarios while reducing computational overhead. The modifications proposed and presented in this paper aim to streamline the process, making it both more efficient and practical for large-scale datasets.

The key adaptations and enhancements to Soroudi's [18] original approach are summarised as follows:

Handling non-uniformly distributed location points: The original method assumes a uniform spatial distribution of points, which does not accurately reflect real-world scenarios, in which location points are typically skewed. In contrast, the proposed approach accommodates non-uniform distributions, ensuring greater practical applicability.

Partitioning strategy – adoption of quadtree partitioning: Instead of using a uniform grid with equal-sized partitions, the proposed approach applies a quadtree partitioning technique. This method allows for adaptive partitioning, in which densely populated regions are assigned more partitions while sparsely populated areas have fewer partitions. Furthermore, the number of partitions can be adjusted dynamically through visual inspection or by modifying the stopping criterion of the quadtree construction, thereby improving flexibility and alignment with real-world location distributions.

Optimised partition evaluation sequence: The sequence in which partitions are evaluated has been refined using a depth-first tree search to identify an optimal circular route among partition midpoints. The original framework relied on a randomly determined circular route, assuming equal-sized partitions with equal distances between midpoints. The proposed method ensures a more structured and efficient traversal of partitions that is aligned with the different sized partitions created by the quadtree technique.

Efficient establishment of optimal connections between partitions: The time required to determine optimal connections between successive partitions has been significantly reduced by leveraging the principles of the triangle inequality. This eliminates the need for an exhaustive pairwise comparison of all points, as was required in the original approach, thereby improving computational efficiency.

Path optimisation in each partition: A two-step heuristic is introduced to enhance the quality of paths in each partition. First, a low-cost heuristic is used to generate an initial feasible path, which is then refined using an additional heuristic (2-opt heuristic) to eliminate path crossings and minimise overall path length.

Incorporation of a real-world case study: A significant extension of the original framework is the inclusion of a real-world case study, which was not previously considered. This case study demonstrates the practical effectiveness of the proposed approach and establishes a benchmark that could serve as a reference for future research in this domain.

Finally, it is important to note that the individual heuristics and algorithms used in this paper may be replaced by any suitable solution strategy. For example, the use of a depth-first search to determine the optimal partition sequence for evaluation could be replaced by a more time-efficient algorithm. The use of a depth-first search method was part of the exploratory and experimental nature of the study, and was not as time-efficient as the other steps in the proposed methodology. Overall, the results indicated that computational time was not a significant constraint in the proposed partitioning framework, and the processing time of individual steps was minimal (generally less than 40 seconds for each step, with some steps only requiring two or three seconds).

In summary, the proposed modifications enhance both the efficiency and the applicability of the partitioning approach, making it more suitable for real-world applications while substantially reducing computational complexity.

6. CONCLUSION

Set partitioning is a well-established and widely applied approach to addressing computationally intensive combinatorial optimisation problems. In this paper, a six-step set partitioning approach, adapted from an existing technique in the literature, was proposed with the objective of generating high-quality approximate solutions. The approach involves partitioning a large dataset and then evaluating each partition in an optimally determined sequence to construct an overall approximated solution. To illustrate the effectiveness of the proposed method, a large-scale real-world instance of the travelling salesman problem with 2 508 nodes was used. The results demonstrate that the methodology, which is easy to implement, effectively reduces the problem's complexity and yields satisfactory solutions to the large TSP instance.

The standard TSP used in this paper is a 'static' problem, while real-world applications are often dynamic. This should create some opportunities for further research. For example, the robustness of the proposed method may be investigated further to include dynamic and stochastic TSP models. Multi-objective optimisation may also enhance path-finding algorithms by implementing two objectives such as minimising distance and maximising safety (e.g., avoiding high-traffic areas). To improve scalability and benchmarking, much larger case studies may be considered.

REFERENCES

[1] **Weinand, J.M., Sorenson, K., Segundo, P.S., Kleinebrahm, M. & McKenna, R.** 2022. Research trends in combinatorial optimisation. *International Transactions in Operational Research*, 29, pp 667-705.

[2] **Mazyavkina, N., Sviridov, S., Ivanov, S. & Burnaev, E.** 2021. Reinforcement learning for combinatorial optimisation: A survey. *Computers and Operations Research*, 134, 105400.

[3] **Yakovlev, S., Kartashov, O. & Pichugina, O.** 2019. Optimisation on combinatorial configurations using genetic algorithms. In Luengo *et al.* (eds), *Proceedings of the Second International Workshop on Computer Modelling and Intelligent Systems*, Vol. 2353.

[4] **Delahaye, D., Chaimatanan, S. & Mongeau, M.** 2019. Simulated annealing: From basics to applications. In Gendreau, M. & Potvin, J.Y. (eds), *Handbook of metaheuristics*, 272, Springer, pp 1-35.

[5] **Vesselinova, N., Steinert, R., Perez-Ramirez, D.F. & Boman, M.** 2020. Learning combinatorial optimisation on graphs: A survey with applications in networking. *IEEE Access*, 8, pp 120388-120416.

[6] **Singh, G. & Rizwanullah, M.** 2022. Combinatorial optimisation of supply chain networks: A reprospective and literature review. *Materials Today: Proceedings*, 62(3), pp 1636-1642.

[7] **Zhang, C., Wu, Y., Ma, Y., Song, W., Le, Z., Cao, Z. & Zhang, J.** 2023. A review on learning to solve combinatorial problems in manufacturing. *IET Collaborative Intelligent Manufacturing*, 5(1), e12072.

[8] **Feng, L., Huang, Y., Zhou, L., Zhong, J., Gupta, A., Tang, K. & Tan, K.C.** 2021. Explicit evolutionary multitasking for combinatorial optimisation: A case study on capacitated vehicle routing problem. *IEEE Transactions on Cybernetics*, 51(6), pp 3143-3156.

[9] **Yang, X., Wang, Z., Zhang, H., Ma, N., Yang, N., Liu, H., Zhang, H. & Yang, L.** 2022. A review: Machine learning for combinatorial optimisation problems in energy areas. *Algorithms*, 15(6), 205.

[10] **Li, K.** 2023. Heuristic task scheduling on heterogeneous UAVs: A combinatorial optimisation approach. *Journal of Systems Architecture*, 40, 102895.

[11] **Naseri, G. & Koffas, M.A.G.** 2020. Application of combinatorial optimisation strategies in synthetic biology. *Nature Communications*, 11, 2446.

[12] **An, Y., Wang, X., Zhu, X., Jiang, S., Ma, X., Cui, J. & Qu, Z.** 2022. Application of combinatorial optimisation algorithm in industrial robot hand eye calibration. *Measurement*, 202, 111815.

[13] **Khumalo, M.T., Chieza, H.A., Prag, K. & Woolway, M.** 2022. An investigation of IBM quantum computing device performance on combinatorial optimisation problems. *Neural Computing and Applications*, 37, pp 611–626.

[14] **Marsten, R.E.** 1972. *An algorithm for large set partitioning problems.* Discussion Paper No. 8, Northwestern University, Kellogg School of Management, Center for Mathematical Studies in Economics and Management Science, Evanston, IL.

[15] **Grenouilleau, F., Legrain, A., Lahrichi, N. & Rousseau, L.** 2019. A set partitioning heuristic for the home health care routing and scheduling problem. *European Journal of Operational Research*, 275(1), pp 295-303.

[16] **Wu, H., Knottenbelt, W.J. & Wolter, K.** 2019. An efficient application partitioning algorithm in mobile environments. *IEEE Transactions on Parallel and Distributed Systems*, 30(7), pp 1464-1480.

[17] **Wu, C., Kamar, E. & Horvitz, E.** 2016. Clustering for set partitioning with a case study in ridesharing. *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp 1384-1388.
[18] **Soroudi, A.** 2024. Solving large scale TSP using OR. *LinkedIn*. Available at: https://www.linkedin.com/pulse/solving-large-scale-tsp-using-alireza-soroudi-3j5qe/?trackingId=jExsq9SuQrCCAVbjnUTZqQ%3D%3D. Date accessed: 20 November 2024.
[19] **Kaya, E., Gorkemli, B., Akay, B. & Karabogo, D.** 2022. A review on the studies employing artificial bee colony algorithm to solve combinatorial optimisation problems. *Engineering Applications of Artificial Intelligence*, 115, 105311.
[20] **Sanchez, M., Cruz-Duarte, J.M., Ortiz-Bayliss, J.C., Ceballos, H., Terashima-Marin, H. & Amaya, I.** 2020. A systematic review of hyper-heuristics on combinatorial optimisation problems. *IEEE Access*, 8, pp 128068-128095.
[21] **Rahman, M.A., Sokkalingam, R., Othman, M., Biswas, K., Abdullah, L. & Kadir, E.A.** 2021. Nature-inspired metaheuristic techniques for combinatorial optimisation problems: Overview and recent advances. *Mathematics*, 9, 2633.
[22] **Radhakrishnan, A. & Jeyakumar, G.** 2021. Evolutionary algorithm for solving combinatorial optimisation: A review. In Saini, H.S., Sayal, R., Govardhan, A. & Buyya, R. (eds), *Innovations in Computer Science and Engineering*. Lecture Notes in Networks and Systems, Springer, 171, pp 539-545.
[23] **Gen, M. & Lin, L.** 2023. Genetic algorithms and their applications. In Pham, H. *(ed.)*, *Handbook of engineering statistics*, Springer, pp 635- 674.
[24] **Mingozzi, A., Boschetti, M.A, Ricciardelli, S. & Bianco, L.** 1999. A set partitioning approach to the crew scheduling problem. *Operations Research*, 47(6), pp 873-888.
[25] **Cattrysse, D.G., Salomon, M. & Van Wassenhove, L.N.** 1994. A set partitioning heuristic for the generalised assignment problem. *European Journal of Operational Research*, 72(1), pp 167-174.
[26] **Matatov, N., Rokach, L. & Maimon, O.** 2010. Privacy-preserving data mining: A feature set partitioning approach. *Information Sciences*, 180(14), pp 2696-2720.
[27] **Borisovsky, P.A., Delorme, X. & Dolgui, A.** 2013. Balancing reconfigurable machining lines via a set partitioning model. *International Journal of Production Research*, 52(13), pp 4026–4036.
[28] **Michalak, T., Rahwan, T., Elkind, E., Wooldridge, M. & Jennings, N.R.** 2016. A hybrid exact algorithm for complete set partitioning. *Artificial Intelligence*, 230, pp 14-50.
[29] **Lewis, A., Kochenberger, G. & Alidaee, B.** 2008. A new modelling and solution approach for the set-partitioning problem. *Computers & Operations Research*, 35(3), pp 807-813.
[30] **Laporte, G.** 2009. Fifty years of vehicle routing. *Transportation Science*, 43(4), pp 408-416.
[31] **Agarwal, Y.K., Mathur, K. & Salkin, M.E.** 1989. A set-partitioning-based exact algorithm for the vehicle routing problem. *Networks*, 19, pp 731-749.
[32] **Bulbul, K.G. & Kasimbeyli, R.** 2025. A new mathematical model and solution method for the asymmetric traveling salesman problem with replenishment arcs. *Applied Mathematics and Computation*, 494, 129278.
[33] **Gao, Y., Wang, Y. & Pei, Z.** 2012. An improved particle swarm optimisation for solving generalised travelling salesman problem. *International Journal of Computing Science and Mathematics*, 3(4), pp 385-393.
[34] **Bektas, T.** 2006. The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega*, 34(3), pp 209-219.
[35] **Sun, L., Karwan, M.H. & Diaby, M.** 2018. The indefinite period traveling salesman problem. *European Journal of Operational Research*, 270(3), pp 1171-1181.
[36] **Veenstra, M., Roodbergen, K.J., Vis, I.F.A. & Coelho, L.C.** 2017. The pickup and delivery traveling salesman problem with handling costs. *European Journal of Operational Research*, 257(1), pp 118-132.
[37] **Henchiri, A., Bellalouna, M. & Khasnaji, W.** 2014. Probabilistic traveling salesman problem: A survey. Federated Conference on Computer Science and Information Systems, *Annals of Computer Science and Information Systems*, 3, pp 55-60.
[38] **Salman, R., Ekstedt, F. & Damaschke, P.** 2020. Branch-and-bound for the precedence constrained generalized traveling salesman problem. *Operations Research Letters*, 48(2), pp 163-166.
[39] **Dash, S., Günlük, O., Lodi, A. & Tramontani, A.** 2011. A time bucket formulation for the traveling salesman problem with time windows. *INFORMS Journal on Computing*, 24(1), pp 132-147.
[40] **Monnot, J., Paschos, V. & Toulouse, S.** 2003. Approximation algorithms for the traveling salesman problem. *Mathematical Methods of OR*, 56, pp 387–405.
[41] **Laporte, G.** 1992. The travelling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59, pp 231-247.
[42] **Larsen, J.** n.d. *Set partitioning and applications*. 42134 Advanced Topics in Operations Research, Department of Management Engineering, Technical University of Denmark. Available at: https://www2.imm.dtu.dk/courses/02735/sppintro.pdf Date accessed: 6 January 2025.

[43] **Dantzig, G.B., Fulkerson, D.R. & Johnson, S.M.** 1954. Solution of a large-scale travelling-salesman problem. *Operations Research*, 2, pp 393-410.
[44] **Miller, C.E., Tucker, A.W. & Zemlin, R.A.** 1960. Integer programming formulation of travelling salesman problems. *Journal of the Association for Computing Machinery*, 7, pp 326-329.
[45] **Xu, G., Zhou, J. & Shu Q.** 2022. Application of quadtree decomposition to intersections search of air-sea gravity survey grid. *Geomatics and Information Science of Wuhan University*, 47(11), pp 1847-1853.
[46] **Revanna, C.R. & Keshavamurthy, C.** 2020. A new partial image encryption method for document images using variance based quad tree decomposition. *Journal of Electrical and Computer Engineering*, 10(1), pp 786-800.
[47] **Jewsbury, R., Bhalerao, A. & Rajpoot, N.M.** 2021. A quadtree image representation for computational pathology. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp 648-656.
[48] **Uddin, F., Riaz, N., Manan, A., Mahmood, I., Song, O-Y., Malik, A.J. & Abbasi, A.A.** 2023. An improvement to the 2-Opt heuristic algorithm for approximation of optimal TSP tour. *Applied Sciences*, 13, 7339.
[49] **Luger, G.F.** 2002. *Artificial intelligence: Structures and strategies for complex problem solving*, 4th ed. Harlow: Addison Wesley, Pearson Education Limited.
[50] **Christofides, N.** 2022. Worst-case analysis of a new heuristic for the travelling salesman problem. *Operations Research Forum*, 3, 20.
[51] **Andrews, J. & Sethian, J.A.** 2007. Fast marching methods for the continuous traveling salesman problem. *Proceedings of the National Academy of Science (PNAS)*, 104(4), pp 1118-1123.