RAIRO-Oper. Res. 59 (2025) 1035–1065
https://doi.org/10.1051/ro/2024140

RAIRO Operations Research
www.rairo-ro.org

SOLUTION APPROACHES TO THE THREE-INDEX ASSIGNMENT PROBLEM

Mohamed Mehbali*

Abstract. This paper delves into the axial Three-Index Assignment Problem (3IAP), alternatively known as the Multi-Dimensional Assignment Problem, defined as an extension of the classical two-dimensional assignment problem. 3IAP entails allocating n tasks to n machines in n factories, ensuring one task is completed by one machine in one factory at a minimum total cost. This combinatorial optimization problem is classified as *NP*-hard due to its inherent complexity and being the subject of much scholarly research and investigation. The study adopts an algorithmic approach to devise rapid and effective solutions for 3IAP. A new heuristic Greedy-Style Procedure (GSP) is introduced for solving 3IAP, achieving feasible solutions within polynomial time. Particular configurations of cost matrices has allowed to reach quality solutions. Examining tie-cases and matrix ordering unveiled innovative variants. Further investigation of cost matrix attributes facilitates the development of two new heuristic categories, offering optimal or nearly optimal solutions for 3IAP. Extensive numerical experiments validate the effectiveness of the heuristics, generating quality solutions in a short computational time. Furthermore, we implement two potent methods using optimization solvers, achieving optimal solutions for 3IAP within competitive CPU times.

Mathematics Subject Classification. 90C27, 90C35, 90C59.

Received March 13, 2023. Accepted July 4, 2024.

1. Introduction

This study explores an optimization problem, specifically the one-to-one allocation of tasks to agents, aiming to minimize incurred costs. Recognized widely in scholarly literature, the problem is known as the two-dimensional assignment problem or Linear Assignment Problem (LAP), a fundamental model in combinatorial optimization. LAP possesses both theoretical significance and broad applicability, for example, optimizing staff deployment, allocating tasks to machines, and teacher-class assignments for enhanced performance, productivity, and targeted educational objectives. This optimization problem is vital, offering valuable solutions for numerous real-life applications.

Considering further location constraints, resource availability, and timetabling constraints within assignment problems transforms them into three-index assignment problems (3IAPs). The introduction of a *'third dimension'* makes the problem harder. The Multi-Index Assignment Problem (MAP) is a natural extension of LAP. Despite extensive research conducted over the past five decades, MAP remains one of the most challenging combinatorial problems to solve optimally.

A polynomial-time algorithm already exists for solving the standard LAP. An efficient algorithm known for LAP is the Hungarian method [34, 42] with computational complexity $O(n^3)$. However, there is no polynomial-

Keywords. Multi-Index assignment problem, Hungarian method, heuristics, combinatorial optimization.

CRIT, London South Bank University, 103 Borough Road, London SE1 0AA, UK.
*Corresponding author: mehbalim@sbu.ac.uk

time algorithm for 3IAP solution due to its intractability. In fact, the MAP is classified *NP*-hard for any dimension greater than or equal to 3 [18]. Although 3IAP is easy to define and illustrate, the problem turns out hard to solve optimally in a reasonable execution time.

Numerous studies have been undertaken over the past half-century, but 3IAP still remains one of the most challenging combinatorial problems to find an optimal solution for. Only small-sized instances can be solved optimally. Hence, only approximate solutions are to be expected for larger instances in a reasonable time. From an algorithmic approach, the paper aims to devise new efficient heuristics for solving 3IAP outperforming existing methods. This improvement is attainable by achieving better solutions, saving computational time, or increasing the problem size.

This paper is divided into six sections. Section 1 introduces and describes the assignment problem, and Section 2 provides some real-world applications and project motivation. Section 3 covers the literature review to date. Next Section 4 presents two mathematical formulations of the problem as an integer linear programmes. Section 5 describes two heuristics illustrated by examples and discusses the tie-cases from which three variants derive. Then two heuristic categories are proposed, tested and two exact solution algorithms are implemented. Finally Section 6 concludes the paper along with an overall summary.

2. Applications

Both 3IAP and MAP have diverse real-world applications. The most studied case of MAP in the last three decades is 3IAP which has received more attention and has seen its applications extended to higher dimensions in various fields. Typical MAP applications are found in data association [48,51], education timetabling, scheduling, satellite launching and biology.

Pierskalla [47] presented one of the first applications of MAP to dynamic facility locations, showing how to schedule capital investments, over specific time periods, while Grundel *et al.* [20] examined MAP application to production planning and resource allocation.

The literature contains further applications, such as satellite launching [35], multisurveillance [6, 47], multi-target tracking [57], radar surveillance, such as target tracking systems, image recognition and air traffic control [20]. A recent study [63] introduced a multi-target tracking approach utilizing a multi-sensor labelled multi-Bernoulli filter within underwater multi-static networks, employing autonomous underwater vehicles. Other examples include computer vision [60] or circuit board assembly [54].

Another popular application is education timetabling, teachers' allocation to courses and time sessions [45,53]. Also Wulan *et al.* [61] studied how to determine optimal solutions in learning outcomes. Frieze and Yadegar [16] applied 3IAP to schedule teaching practice for teacher trainees. MAP also applies to healthcare sector timetabling and local daily care centres for the elderly [15].

Applications of 3IAP are also encountered in biomedical. For instance, [4] used a 3IAP model to solve three protein-interaction networks' alignment while Gabrovšek *et al.* [17] investigated the multi-association of three sets of interactions (diseases, drugs and genes). The problem was formulated as a 3IAP model then solved by the multiple Hungarian method.

3. Mathematical formulations

Two formulations are presented below, one using a system of linear equations while the other in matrix form. Each of these formulations will later be implemented in a computer program.

3.1. First formulation

The earliest mathematical formulation of the axial 3IAP is the following integer linear program LP1 [46]. Given three disjoint sets $I, J,\ K$ and a cost function $c : I \times J \times K \longrightarrow \mathbb{R}_+$, find a collection of triplets $S \subset I \times J \times K$, such that each element from each set appears in exactly one triplet. The nonnegative coefficient

c_{ijk} represents the cost of assigning job $i \in I$ to machine $j \in J$ in factory $k \in K$. The decision variable (1e), $x_{ijk} = 1$ if the job i is assigned to machine j in factory k, and 0 otherwise.

The following model LP1 represents 3IAP.

$$\text{Minimize} \qquad Z = \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} c_{ijk}x_{ijk} \tag{1a}$$

$$\text{subject to:} \qquad \sum_{j=1}\sum_{k=1} x_{ijk} = 1, \qquad i \in I \tag{1b}$$

$$\sum_{i=1}\sum_{k=1} x_{ijk} = 1, \qquad j \in J \tag{1c}$$

$$\sum_{i=1}\sum_{j=1} x_{ijk} = 1, \qquad k \in K \tag{1d}$$

$$x_{ijk} \in \{0,1\}, \qquad i \in I, j \in J, k \in K \tag{1e}$$

Linear equalities (1b)–(1d) define the problem constraints, while the objective function (1a) Z is defined by a $\mathbb{R}^{n^3}$-vector, evaluates the assignment total cost, and Z should be minimum.

The model represents an integer linear programme with n^3 variables. If the sets I, J, and K have different cardinalities, 3IAP is considered unbalanced, in this case simply add zero cost to the rows and columns as appropriate to balance the problem. A balanced 3IAP model resembles LP1 with the supplementary condition on cardinalities $|I| = |J| = |K| = n$.

3.2. Second formulation

Another mathematical formulation LP2 of 3IAP, is expressed in matrix form:

$$\text{Minimize} \qquad Z = C \cdot x \tag{2a}$$

$$\text{subject to:} \qquad A_n \cdot x = \boldsymbol{e}, \tag{2b}$$

$$x \in \{0,1\}^{n^3} \tag{2c}$$

For convenience, we assume $n \geq 3$. The objective function is the scalar product $Z = C \cdot x$ (2a) equal to the assignment total cost, where C is an n^3-dimensional row vector with nonnegative components, and the decision variable $x \in \{0,1\}^{n^3}$ (2c). In (2b), A_n represents the constraint matrix whose coefficients are in {0, 1}. Moreover, A_n is a sparse matrix of rank $(3n-2)$ with a special structure. In the right-hand side of (2b) the column vector $\boldsymbol{e}$ has $3n$ components, all equal to 1. If the constraint (1e) or (2c) is relaxed, the resulting linear program optimum constitutes a lower bound for the model.

All 3IAPs of size n, share the same constraint matrix A_n, distinct 3IAPs differ only in their objective functions. Note that the objective function can be formulated by n square matrices, each of which is associated with a factory. Investigating the function objective characteristics for solution approaches would be worthwhile. The above integer linear program holds up to $(n!)^2$ feasible solutions. 3IAP remains one of the most exacting combinatorial problems to solve optimally since Karp [31] proved that MAP is *NP*-hard for any dimension greater than or equal to 3.

4. Literature review

The literature comprises numerous studies carried out over the last five decades. Among the earliest solution methods is the tri-substitution heuristic proposed by [46] to solve 3IAP. The mainstream methods for solving MAP can be summarised in six categories.

4.1. Approximation algorithms

Spieksma [54] defined two essential concepts which help in the understanding of the approximation algorithm's notion. Prior [10] examined special cases of 3IAP where the edge lengths satisfy the triangular inequality. Although both versions of 3IAP are proved *NP*-hard, the authors developed heuristics that always provide a feasible solution, whose objective function value for such solutions is within 3/2 the optimum for one and 4/3 the other. Spieksma and Woeginger [55] studied two further geometric cases of 3IAP where the three sets I, J and K of n points each, are considered in the Euclidean plane.

Later, Johnsson *et al.* [24] explored the three-matching problem in the Euclidean plane and identified it as a variant of the geometric case of 3IAP. The authors searched, in the Euclidean plane, for a partition of the n points into p subsets, then they developed heuristics based on tabu-search, simulated annealing, and genetic algorithms.

4.2. Exact algorithms

The exact algorithms applied a common branching rule based on fixing one variable at a time to 0 or 1, whereas Balas and Saltzman [8] proposed a new branching strategy by fixing several variables at once at each branching node. Then by incorporating a class of facet inequalities at each branching node, they have successfully solved to optimality 3IAP, up to size $n = 26$, [8].

Burkard *et al.* [9] investigated various Branch and Bound (B&B) schemes for 3IAP. In particular, they experimented with several branching rules by utilizing the results on the three-index assignment polytope, previously studied by [7]. Pasiliao *et al.* [44] described two B&B algorithms for solving MAP based on different tree representations. One noted that their algorithm performance's depends on the size and dimension of the problem.

4.3. Lagrangian relaxation

The common technique used in integer linear programming is to relax the integrity constraint. Then the optimal solution to the resulting linear program usually provides a valid lower bound on the original problem.

The Lagrangian relaxation is a popular technique widely used in the literature [19, 48–50, 62]. From 3IAP formulation LP1, a new Lagrangian relaxation is obtained whenever one or two of three constraint sets are inserted into the objective function with associated multipliers. Frieze and Yadegar [16] developed for 3IAP a heuristic using Lagrangian relaxation. The authors after relaxing the LP1 constraint (1e), found a feasible solution.

The key advantage of the Lagrangian relaxation method is that it permits the computation of both upper and lower bounds on the optimal solution. The method thus enables the evaluation of the quality of the solution obtained. [8] applied four variants of Lagrangian relaxation to 3IAP. Recently, Li *et al.* [36] considered a particular type of the Lagrangian method where they relaxed the three sets of constraints of LP1. This approach allows setting a sufficient and necessary condition for the duality gap to be zero. The authors noted that their approach could reach an optimal solution to the assignment problem. Dokka *et al.* [11] investigated another type called surrogate relaxation as a metaheuristic.

4.4. Polyhedral approach

The polyhedral approach is demonstrated through numerous studies undertaken on the axial 3IAP. Euler [14] is the first to investigate the structure of the convex hull of 3IAP feasible solutions and the role of the odd cycle in a class of the polyhedron facets. [7] studied the three-index assignment polytope facial structure intensively, and identified several facet classes.

After a follow-up study, Balas and Qi [5] proposed a separation procedure for some classes of facets previously identified, along with a new facet class. Qi and Sun [52] summarised the major results achieved on the polyhedral structure of 3IAP polytope as they described several facet classes and provided separation algorithms for them. Later, Dokka *et al.* [13] defined a novel class of facets then Dokka and Spieksma [12] studied the facets of the 3IAP polytope. For a detailed overview of the approach, the reader can refer to the following studies [38, 39].

Appa *et al.* [3] expanded the study to the polytope of the MAP in a general context. Their work unified and generalized earlier results; in particular, they set the dimensions of both axial and planar MAP polytopes.

Furthermore, they identified a novel class of clique facets for the axial MAP polytopes and established a necessary condition for the existence of MAP solutions.

However, Kravtsov [32] investigated 3IAP polytope from a different perspective and examined the structure of non-integer vertices of the polytope, then [33] characterized such vertices and studied their combinatorial properties.

4.5. Local search method

The local search method constitutes another category of heuristics designed to solve *NP*-hard combinatorial optimization problems. A local search method is an iterative procedure that continues running until an optimal solution is found, or up to a fixed number of iterations. The so-called neighbourhood-centred methods attempts to explore the neighbourhoods of the current solution. The literature includes various local research methods that consider diverse classes and neighbourhoods' sizes [29, 30].

Kammerdiner *et al.* [28], prior [26, 27] described a very large-scale neighbourhood search method for MAP. Perez [45] studied intensively MAP and proposed a new metaheuristic by combining a local search heuristic with the evolutionary algorithm.

4.6. Genetic algorithm

Moreover, genetic (or evolutionary) algorithms are inspired by nature selection and are commonly used to generate good solutions to optimization problems. The implementation of genetic algorithms relies on three operators, namely, mutation, crossover and selection similar to those encountered in biology. In this regard, several studies have been carried out on MAP [23,37,58,59]. Pal *et al.* [43] introduced a MAP heuristic combining genetic algorithms with particle swarm optimization, although with limited numerical testing.

4.7. Other methods

He *et al.* [22] developed a sophisticated heuristic called Approximate Muscle guided Beam Search to balance solution quality with execution time. Probabilistic approaches were proposed for solving the axial 3IAP. [40] rather than optimize the objective function, minimized its expectation. On the other hand, Afraimovich and Emelin [1, 2] combined pairs of feasible solutions attempting to find an optimal solution. If the induced graph is disconnected, their algorithm can find a better solution when it exists; otherwise, there is no improvement. This algorithm can be applied as a supplement to heuristics for post-processing 3IAP approximate solutions. Then, the authors [2] discussed the computational complexity of such combinations.

Our paper envisages an original approach by tackling the problem from an algorithmic perspective. Two fast heuristic classes are proposed to achieve approximate solutions, then two computer programs are implemented for solving optimally 3IAPs, followed by numerical experiments.

5. New heuristics for 3IAP

This section delineates the development of new heuristics for 3IAP, initiating with an efficient algorithm devised for the swift attainment of feasible solutions, followed by two variants employing pre-sorted matrices. After matrix attribute analysis, two heuristic categories are proposed for solving 3IAP.

5.1. Greedy-Style procedure

The Greedy-Style Procedure (GSP) introduced for the first time, is a heuristic method for 3IAP, consistently yielding feasible solutions.

Let $\mathcal{M}^{n^2}$ denote the set of all $n \times n$-matrices. Let $I = J = K = \{1, 2, \ldots, n\}$ be, for convenience, three sets of n elements, assuming $n \geq 3$. Clearly the objective function Z can be represented by a sequence of n elements

TABLE 1. Example of 3IAP of size n.

Factories		F_1				F_2				...	F_n			
Machines		m_1	m_2	...	m_n	m_1	m_2	...	m_n	...	m_1	m_2	...	m_n
Jobs	j_1	c_{111}	c_{121}	...	c_{1n1}	c_{112}	c_{122}	...	c_{1n2}	...	c_{11n}	c_{12n}	...	c_{1nn}
	j_2	c_{211}	c_{221}	...	c_{2n1}	c_{212}	c_{222}	...	c_{2n2}	...	c_{21n}	c_{22n}	...	c_{2nn}
	...		...		...		...		...	...		...		...
	j_n	c_{n11}	c_{n21}	...	c_{nn1}	c_{n12}	c_{n22}	...	c_{nn2}	...	c_{n1n}	c_{n2n}	...	c_{nnn}

TABLE 2. Example of 3IAP of size 2.

Factories		F_1		F_2	
Machines		m_1	m_2	m_1	m_2
Jobs	j_1	c_{111}	c_{121}	c_{112}	c_{122}
	j_2	c_{211}	c_{221}	c_{212}	c_{222}

of $\mathcal{M}^{n^2}$ each of which is associated with a factory, as Table 1 shows. Coefficient c_{ijk} for $\forall i, j, k = 1, 2, \ldots, n$ evaluates the cost of assigning job i to machine j in factory k.

The GSP heuristic sequentially processes matrices, starting with C_1, the first matrix in the list, applying the Hungarian algorithm to solve the corresponding LAP. GSP selects the minimal cost c_{pq1}, post-Hungarian method application, after that excising row p and column q from other matrices and removing C_1 from the sequence.

GSP systematically reduces the size of 3IAP by one unit after each iteration. The process is sustained until only two (2×2)-matrices remain, as illustrated in Table 2, which can be easily resolved. A feasible solution for the original assignment problem is derived by adding all selected minima and recovering their corresponding indices.

5.1.1. GSP algorithm

Algorithm 1 describes the sequence of steps of the procedure.

Algorithm 1: GREEDY-STYLE PROCEDURE - GSP

Input: A sequence of cost matrices $\langle C_1, C_2, \ldots, C_n \rangle$
Output: A feasible solution for 3IAP.

```
for k ← n to 2 do
    while k ≥ 3 do
        Apply Hungarian method to C_k.
        From the Hungarian method solution, select the minimum cost, let say c_ijk.
        Store c_ijk and remove matrix C_k.
        Delete row p and column q from all other matrices.
        C_{k-1} ← C_k
    if k = 2 then
        Solve the last 3IAP of size 2;
    Retrieve all previous allocations.
    Sum all selected costs.
return Solution for 3IAP.
```

Theorem 5.1. *Given an instance of 3IAP of size $n \geq 2$, GSP always returns a feasible solution for the problem, in polynomial time.*

Proof. If $n = 2$, it is obvious. Assuming $n \geq 3$, at each iteration $k = n, n-1, \ldots, 3$; GSP applies the Hungarian method to the first matrix in the chosen order. From one iteration to the next, GSP reduces 3IAP size by one unit and the Hungarian method executes $O(n^3)$ operations. As the procedure runs over $n-2$ iterations, thus the overall computational complexity is at most $O(n^4)$, therefore it is polynomial. □

Example 1. 3IAP of size 2.

$$C_1 = \begin{array}{c} \\ j_1 \\ j_2 \end{array}\!\!\begin{array}{c} \begin{array}{cc} m_1 & m_2 \end{array} \\ \begin{bmatrix} c_{111} & c_{121} \\ c_{211} & c_{221} \end{bmatrix} \end{array} \quad \text{and} \quad C_2 = \begin{array}{c} \\ j_1 \\ j_2 \end{array}\!\!\begin{array}{c} \begin{array}{cc} m_1 & m_2 \end{array} \\ \begin{bmatrix} c_{112} & c_{122} \\ c_{212} & c_{222} \end{bmatrix} \end{array}$$

Let C_1 and C_2 be two matrices associated respectively with factory 1 and factory 2; since a job in each factory is performed by one machine, the problem admits four possible solutions from which the optimal solution derives straightforwardly, of cost equal to $\min\{c_{111} + c_{222}, c_{122} + c_{211}, c_{121} + c_{212}, c_{112} + c_{221}\}$.

5.1.2. GSP illustration

Example 2. 3IAP of size 4.

Let C_1, C_2, C_3 and C_4, be cost matrices associated with four factories. Rows correspond to jobs and columns to machines.

$$\mathrm{C}_1 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 87 & 59 & 83 & 41 \\ 09 & 47 & 26 & 59 \\ 20 & 88 & 95 & 10 \\ 53 & 26 & 14 & 75 \end{pmatrix} \end{array} \quad \mathrm{C}_2 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 66 & 45 & 09 & 51 \\ 15 & 69 & 100 & 58 \\ 06 & 84 & 60 & 41 \\ 13 & 12 & 07 & 05 \end{pmatrix} \end{array}$$

$$\mathrm{C}_3 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 17 & 46 & 34 & 99 \\ 20 & 62 & 89 & 04 \\ 27 & 72 & 04 & 100 \\ 45 & 52 & 67 & 94 \end{pmatrix} \end{array} \quad \mathrm{C}_4 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 96 & 33 & 42 & 08 \\ 16 & 55 & 99 & 63 \\ 72 & 60 & 70 & 28 \\ 36 & 96 & 63 & 24 \end{pmatrix} \end{array}$$

Clearly the problem requires 64 variables and holds up to 576 feasible solutions. If $x_{iii} = 1$, for $\forall i = 1, 2, 3, 4$ and 0 otherwise, is a trivial solution, and the value of the objective function $Z = 184$ might constitute an upper bound for 3IAP optimal solution. For any instance of size n, the restriction of the problem to one factory, the allocation of n tasks to n machines within that factory, reduces it to a two-dimensional assignment problem, solvable in polynomial time.

Let us apply Algorithm 1 to the above example.

Iteration 1. $k = 4$.

Starting with C_1 the first matrix, the Hungarian method produces an optimal solution S_1 to LAP related to C_1, as shown below.

$$\mathrm{C}_1 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 87 & 59 & 83 & 41 \\ 09 & 47 & 26 & 59 \\ 20 & 88 & 95 & 10 \\ 53 & 26 & 14 & 75 \end{pmatrix} \end{array} \quad \Rightarrow \quad \mathrm{S}_1 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} - & 59 & - & - \\ (9) & - & - & - \\ - & - & - & 10 \\ - & - & 14 & - \end{pmatrix} \end{array}$$

S_1 represents the Hungarian method solution for C_1.

Step 4: From the solution S_1, select the minimum cost 9 corresponding to the coefficient c_{211}. Thus job 2 will be allocated to machine 1 in factory 1.

Step 5 and 6: Next, delete row 2 and column 1 from matrices C_2, C_3 and C_4, then remove C_1. Hence three submatrices result in.

$$C'_2 = \begin{array}{c} \\ j_1 \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 45 & 09 & 51 \\ 84 & 60 & 41 \\ 12 & 07 & 05 \end{pmatrix} \end{array} \quad C'_3 = \begin{array}{c} \\ j_1 \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 46 & 34 & 99 \\ 72 & 04 & 100 \\ 52 & 67 & 94 \end{pmatrix} \end{array} \quad C'_4 = \begin{array}{c} \\ j_1 \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 33 & 42 & 08 \\ 60 & 70 & 28 \\ 96 & 63 & 24 \end{pmatrix} \end{array}$$

Iteration 2. $k = 3$.

Step 3. Applying the Hungarian method to matrix C'_2.

Step 4 and 5. This results in the LAP optimal assignment associated with factory 2, whose lowest value equals 9, the cost of assigning job 1 to machine 3 in factory 2.

$$\mathrm{S}_2 = \begin{array}{c} \\ j_1 \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} - & (9) & - \\ - & - & 41 \\ 12 & - & - \end{pmatrix} \end{array}$$

The Hungarian method solution S_2.

Step 6. Next, delete the first row and the middle column from submatrices C'_3 and C'_4, then eliminate C'_2. Two new submatrices result in.

Step 8 and 9. The two submatrices are.

$$C''_3 = \begin{array}{c} \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{cc} m_2 & m_4 \end{array} \\ \begin{pmatrix} 72 & 100 \\ (52) & 94 \end{pmatrix} \end{array} \quad \text{and} \quad C''_4 = \begin{array}{c} \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{cc} m_2 & m_4 \end{array} \\ \begin{pmatrix} 60 & (28) \\ 96 & 24 \end{pmatrix} \end{array}$$

Minimum assignment related to C''_3 and C''_4.

The minimum cost 80 is attained when job 3 is assigned to machine 4 in factory 4, and job 4 is assigned to machine 2 in factory 3.

Step 10 and 11. A feasible solution is found for the original 3IAP when all variables $x_{ijk} = 0$, except $x_{132} = x_{211} = x_{344} = x_{423} = 1$, with a total cost of 98.

$$\begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc|cccc|cccc|cccc} m_1 & m_2 & m_3 & m_4 & m_1 & m_2 & m_3 & m_4 & m_1 & m_2 & m_3 & m_4 & m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} - & - & - & - & - & - & 9 & - & - & - & - & - & - & - & - & - \\ 9 & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & 28 \\ - & - & - & - & - & - & - & - & - & 52 & - & - & - & - & - & - \end{pmatrix} \end{array}$$

The GSP solution for 3IAP.

Cost matrices are endowed with a particular data structure, and the GSP heuristic processes them following their entry sequence by the FIFO principle. Reversing the sequence in which cost matrices are processed, could really impact solution quality. Applying the GSP heuristic to the same matrices in reverse order, thereby adhering to the LIFO principle, could produce varied results.

Reversing the order of 3IAP matrices and applying the GSP heuristic can yield an alternative solution distinct from the initial one. Thus, it is advisable to experiment with both sequences, as one might be better than the other. The inquiry examines the effects of modifying matrices' order before applying GSP, which could result in improved solutions.

Let us apply Algorithm 1 to the same example but with the cost matrices arranged in LIFO order. Commencing with matrix C_4, the Hungarian method produces an optimal solution S_4 to LAP related to C_4, as illustrated below:

Iteration 1. $k = 4$.

$$\mathrm{C}_4 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array} \begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 96 & 33 & 42 & 08 \\ 16 & 55 & 99 & 63 \\ 72 & 60 & 70 & 28 \\ 36 & 96 & 63 & 24 \end{pmatrix} \end{array} \quad \Rightarrow \quad \mathrm{S}_4 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array} \begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} - & 33 & - & - \\ (16) & - & - & - \\ - & - & - & 28 \\ - & - & 63 & - \end{pmatrix} \end{array}$$

S_4 represents the Hungarian method solution for C_4.

Step 4: From the solution S_4, select the minimum cost 16 corresponding to the coefficient c_{214}. Thus job 2 will be assigned to machine 1 in factory 4.

Step 5 and 6: Next, delete the second row and first column from matrices C_3, C_2 and C_1, then remove C_4. Thus, three submatrices C_3', C_2', and C_1' result, and the procedure continues until the last iteration.
When Algorithm 1 terminates, a feasible solution is found for the original 3IAP where all variables $x_{ijk} = 0$, except $x_{114} = x_{214} = x_{323} = x_{432} = 1$, with a total cost of 73.

$$\begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array} \begin{array}{c} \begin{array}{cccc|cccc|cccc|cccc} m_1 & m_2 & m_3 & m_4 & m_1 & m_2 & m_3 & m_4 & m_1 & m_2 & m_3 & m_4 & m_1 & m_2 & m_3 & m_4 \end{array} \\ \left(\begin{array}{cccc|cccc|cccc|cccc} - & - & - & 41 & - & - & - & - & - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - & - & - & - & - & 16 & - & - & - \\ - & - & - & - & - & - & - & - & - & 4 & - & - & - & - & - & - \\ - & - & - & - & - & - & 12 & - & - & - & - & - & - & - & - & - \end{array}\right) \end{array}$$

Another solution for 3IAP provided by GSP.

GSP provides two possible solutions for 3IAP, contingent upon the cost matrices' order. While the second result may not surpass the quality of the first, it remains a viable alternative and the best solution should be selected after testing both. The next section will explore the effect of matrix order alteration on the quality of solutions generated by GSP.

5.2. Diagonals method

Kadhem [25] studied 3IAP and introduced an alternative way of sorting cost matrices called the Diagonals Method (DM). The approach organises the cost matrices in ascending or descending order and the DM heuristic often yields at least two solutions for 3IAP.

Definition 5.2. *Let matrix A be an element of $\mathcal{M}^{n^2}$. The sum of the diagonal (anti-diagonal) coefficients of A is called the '**trace**' ('**anti-trace**' respectively) of A, denoted $tr(A)$ ($atr(A)$ respectively), i.e., $tr(A) = \sum_{i=1}^{n} a_{ii}$ and $atr(A) = \sum_{i=1}^{n} a_{i(n+1-i)}$.*

Definition 5.3. *Let matrix A be an element of $\mathcal{M}^{n^2}$. The '**diagonal factor**' of A, denoted $df(A)$, is the maximum of the trace and anti-trace of A, i.e. $df(A) = Max(tr(A), atr(A))$.*

The diagonal factor of a matrix A, is a new concept introduced for the first time. The diagonal factor plays a pivotal role in the DM heuristic, since it serves to arrange the cost matrices. The DM heuristic, structured in two phases, aims to solve 3IAP. The initial phase involves sorting the cost matrices in ascending or descending order, followed by a second which invokes the GSP procedure.

5.2.1. DM illustration

Example 3. Let us re-examine the previous case of 3IAP of size 4.
C_1, C_2, C_3 and C_4 being four matrices of $\mathcal{M}^{4^2}$ associated with four factories.

$$\mathrm{C}_1 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array} \begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 87 & 59 & 83 & 41 \\ 09 & 47 & 26 & 59 \\ 20 & 88 & 95 & 10 \\ 53 & 26 & 14 & 75 \end{pmatrix} \end{array} \quad \mathrm{C}_2 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array} \begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 66 & 45 & 09 & 51 \\ 15 & 69 & 100 & 58 \\ 06 & 84 & 60 & 41 \\ 13 & 12 & 07 & 05 \end{pmatrix} \end{array}$$

$$\mathrm{C}_3 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array} \begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 17 & 46 & 34 & 99 \\ 20 & 62 & 89 & 04 \\ 27 & 72 & 04 & 100 \\ 45 & 52 & 67 & 94 \end{pmatrix} \end{array} \quad \mathrm{C}_4 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array} \begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 96 & 33 & 42 & 08 \\ 16 & 55 & 99 & 63 \\ 72 & 60 & 70 & 28 \\ 36 & 96 & 63 & 24 \end{pmatrix} \end{array}$$

Phase 1.

Compute the diagonal factors of the matrices. $df(C_1) = 304$, $df(C_2) = 248$, $df(C_3) = 305$ and $df(C_4) = 245$. The matrices arranged in ascending order of their diagonal factors, are C_4, C_2, C_1, and C_3.

Phase 2.

Iteration 1. $k = 4$.

Step 3. Starting with C_4 the first matrix, the Hungarian method produces an optimal solution S_4 to LAP corresponding to C_4, as shown below:

$$\mathrm{C}_4 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array} \begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 96 & 33 & 42 & 08 \\ 16 & 55 & 99 & 63 \\ 72 & 60 & 70 & 28 \\ 36 & 96 & 63 & 24 \end{pmatrix} \end{array} \Rightarrow \quad \mathrm{S}_4 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array} \begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} - & 33 & - & - \\ (16) & - & - & - \\ - & - & - & 28 \\ - & - & 63 & - \end{pmatrix} \end{array}$$

S_4 represents the Hungarian method solution for C_4.

Step 4: From the solution S_4, select 16 as the minimum cost, corresponding to the coefficient c_{214}. Thus job 2 will be assigned to machine 1 in factory 4.

Step 5 and 6: Next, delete row 2 and column 1 from matrices C_2, C_1 and C_3, then remove C_4. Then three submatrices C_2', C_1' and C_3' result in, and the algorithm continues. Hence, a feasible solution is achieved for the original 3IAP where all variables are $x_{ijk} = 0$, except $x_{132} = x_{214} = x_{341} = x_{423} = 1$, with a total cost of 87.

$$\begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array} \begin{array}{c} \begin{array}{cccc|cccc|cccc|cccc} m_1 & m_2 & m_3 & m_4 & m_1 & m_2 & m_3 & m_4 & m_1 & m_2 & m_3 & m_4 & m_1 & m_2 & m_3 & m_4 \end{array} \\ \left(\begin{array}{cccc|cccc|cccc|cccc} - & - & - & - & - & - & 9 & - & - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - & - & - & - & - & 16 & - & - & - \\ - & - & - & 10 & - & - & - & - & - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - & - & 52 & - & - & - & - & - & - \end{array}\right) \end{array}$$

The DM solution for 3IAP.

Adopting a methodology similar to FIFO and LIFO rules, an alternative solution is anticipated upon reversing the matrices' arrangement or sorting them in descending order of their diagonal factors. In the following, the problem will be re-addressed with cost matrices sorted in descending order according to their diagonal factors, *i.e.*, C_3, C_1, C_2 and C_4.

The Hungarian method applied to matrix C_3 yields optimal solution S_3, highlighting a duplicated minimum cost of 4. The case will be discussed later in further detail. In this iteration, the coefficient c_{243}, the cost for allocating job 2 to machine 4 in factory 3, will be selected as shown below.

$$\mathrm{C}_3 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array} \begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 17 & 46 & 34 & 99 \\ 20 & 62 & 89 & 04 \\ 27 & 72 & 04 & 100 \\ 45 & 52 & 67 & 94 \end{pmatrix} \end{array} \Rightarrow \quad \mathrm{S}_3 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array} \begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \begin{pmatrix} 17 & - & - & - \\ - & - & - & (4) \\ - & - & 4 & - \\ - & 52 & - & - \end{pmatrix} \end{array}$$

The Hungarian method solution S_3.

Step 5 and 6. Delete matrix C_3 and eliminate row 2 and column 4 from all other matrices, and continue executing the DM algorithm until only two submatrices remain. Therefore, a second feasible solution is

yielded for the original 3IAP when all variables $x_{ijk} = 0$, except $x_{124} = x_{243} = x_{312} = x_{431} = 1$, with a total cost of 57, as displayed below.

$$\begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \left(\begin{array}{cccc|cccc|cccc|cccc} - & - & - & - & - & - & - & - & - & - & - & - & - & 33 & - & - \\ - & - & - & - & - & - & - & - & - & - & - & 4 & - & - & - & - \\ - & - & - & - & 6 & - & - & - & - & - & - & - & - & - & - & - \\ - & - & 14 & - & - & - & - & - & - & - & - & - & - & - & - & - \end{array}\right) \end{array}$$

A second DM solution for the same 3IAP.

DM also provides at least two feasible solutions for 3IAP based on the order of the cost matrices. The preferable outcome for the assignment problem is the one with the lowest cost.

5.2.2. DM and tie-cases

Definition 5.4. *At step 4 of Algorithm 1, if two or more coefficients are equal to the minimum cost, the situation is called a '**tie-case**'.*

When the DM algorithm is applied to the previous example, with cost matrices sorted in descending order of their diagonal factors, a tie-case emerges at the first iteration. If the second cost, evaluated at 4 relative to c_{333} is chosen, then DM will usually produce an alternative solution.

$$\mathrm{S}_3 = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \left(\begin{array}{cccc} 17 & - & - & - \\ - & - & - & 4 \\ - & - & (4) & - \\ - & 52 & - & - \end{array}\right) \end{array}$$

S_3 is the Hungarian method solution for C_3.

After removing matrix C_3 and deleting row 3 and column 3 from matrices C_1, C_2 and C_4 the execution of the DM algorithm progresses until another feasible solution for the original problem is identified. All variables of the solution are $x_{ijk} = 0$, except $x_{144} = x_{211} = x_{333} = x_{422} = 1$, resulting in a total cost of 33.

$$\begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \end{array} \\ \left(\begin{array}{cccc|cccc|cccc|cccc} - & - & - & - & - & - & - & - & - & - & - & - & - & - & - & 8 \\ 9 & - & - & - & - & - & - & - & - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - & - & - & 4 & - & - & - & - & - \\ - & - & - & - & - & 12 & - & - & - & - & - & - & - & - & - & - \end{array}\right) \end{array}$$

Another DM feasible solution for 3IAP.

In conclusion, five feasible solutions for 3IAP are identified; GSP generates two with costs of 98 and 73, whereas DM yields three 87, 57 and 33, with the latter being optimal.

By rearranging cost matrices in ascending then descending order and choosing diverse minima in tie-cases, DM reaches, for the same problem, several feasible solutions and the lowest total cost result is selected. Kadhem [25], however, always picked the first minimum emerging in tie-cases despite the presence of multiple candidates.

Selecting the first minimum in tie-cases does not guarantee the best solution. Explicitly enumerating all candidates is impractical and expensive. Therefore, exploring minimum candidates in tie-cases is advised, since the situation generates various solutions. We propose choosing the first, last, or a random candidate in each iteration, leading to three novel variants of the DM algorithm. When a tie-case occurs during an iteration, especially with large instances of 3IAP, at least three options are available, as illustrated in Figure 1.

The following table summarises the six DM solutions found for the previous example 3IAP instance of size 4.

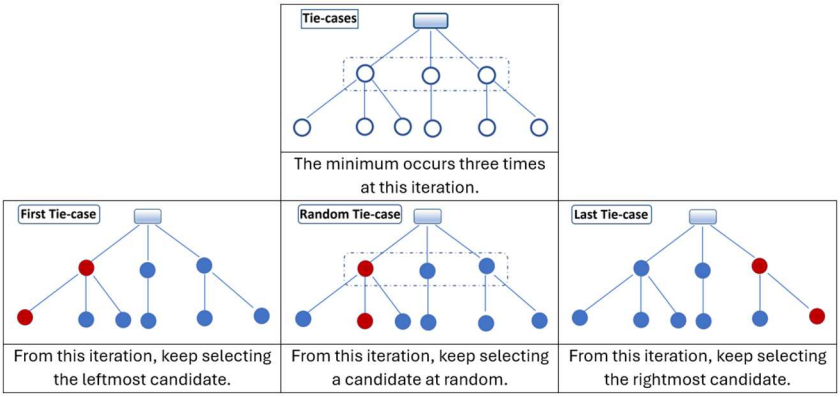

FIGURE 1. Three types of tie-cases.

TABLE 3. DM solutions for Example 3.

	First tie-case		Last tie-case		Random tie	
Sorted in	Cost	Runtime	Cost	Runtime	Cost	Runtime
Ascending	87	0.0312722	87	0.020031	87	0.015622
Descending	57	0.0189335	<u>33</u>	<u>0.015576</u>	33	0.019915

The DM and GSP heuristics generate six potential feasible solutions for each instance by rearranging cost matrices in ascending and descending orders. If multiple results achieve the same total cost, the solution with the lowest execution time is chosen. In Table 3, the execution time is displayed in seconds. For the previous example, DM achieved the minimum cost twice in descending order, the optimal solution is underlined and achieved by selecting the last tie-case. The three variants of each method are considered throughout the rest of the paper.

5.3. Numerical experiments

Computational tests are conducted on a PC with Intel(R) Core(TM) i5-8265U CPU @ 1.60 GHz, and 8.79 GB of RAM; the algorithms are coded in Python 3.9 and run on random samples to test the performance of GSP and DM variants.

To evaluate the performance of the GSP and DM variants, numerical experiments are tested on the same dataset as in [8] which will be referred to as Sample 1 throughout the paper. The random sample consists of 60 instances, generated by the uniform probability distribution from 0 to 100, of size $n = 4, 6, 8, \ldots, 26$, five instances per size. The data and code used in this section are available online in a Github repository: https://github.com/mmehbali/Three-IAP. For each instance the GSP and DM variants provide six feasible solutions, displayed in Tables 4 and 5, respectively; the best solutions found are highlighted.

Analysis of these two tables in Figure 2 shows that GSP's distribution of best solutions found across tie-case types is nearly balanced, with a slim preference for random tie-cases. In contrast, DM exhibits a clear preference for random tie-cases. When partitioning the best solutions found per type of order of cost matrices, the FIFO rule is slightly prevalent for GSP, while DM shows some disparity between the ascending and descending orders.

This examination reveals two key insights; first, the efficacy of random tie-cases is notable and warrants consideration. Secondly, future implementations will prioritize the FIFO rule in the GSP heuristic for unsorted cost matrices, henceforth referred to simply as GSP.

TABLE 4. GSP Variants applied to a data sample from Balas and S., 1991.

		FIFO rule						LIFO rule					
Size	Instance	First tie-case		Last tie-case		Random tie		First tie-case		Last tie-case		Random tie	
		Cost	Runtime	Cost	Runtime	Cost	Runtime	Cost	Runtime	Cost	Runtime	Cost	Runtime
4	1	46	0.00210214	46	0.00320339	46	0.01134753	127	0.00369668	127	0.00471401	127	0.00643301
	2	89	0.00099730	89	0.00365448	89	0.00510907	73	0.00676274	73	0.00338864	73	0.00595522
	3	54	0.00206923	54	0.00194740	54	0.01150131	59	0.00586843	59	0.00860476	59	0.01399684
	4	83	0.00200605	83	0.00820732	83	0.00541782	75	0.00362301	99	0.00890183	75	0.01168871
	5	50	0.00099754	50	0.00199509	50	0.00624371	77	0.00929666	77	0.00801396	77	0.00879145
6	1	73	0.00567222	73	0.01286340	73	0.01871562	89	0.01606989	89	0.01075602	89	0.00666213
	2	51	0.00423217	51	0.00482130	51	0.02397418	70	0.01296616	70	0.00859118	70	0.00562620
	3	101	0.00940967	123	0.00693583	101	0.02210855	140	0.01097131	140	0.01231647	140	0.01776171
	4	113	0.00232983	113	0.00099707	113	0.01994443	121	0.00728488	121	0.01400566	121	0.00970387
	5	73	0.00742221	73	0.00000000	73	0.02569437	88	0.00583696	88	0.01289344	88	0.01687717
8	1	45	0.00316525	45	0.01435137	45	0.01063967	74	0.01742792	74	0.00908208	45	0.03473115
	2	95	0.01857448	105	0.00316381	95	0.00973868	120	0.01691484	66	0.00710893	95	0.03704453
	3	92	0.02015924	65	0.00099921	65	0.02498555	126	0.01836371	88	0.00699759	46	0.04327226
	4	136	0.02007961	136	0.00321698	136	0.01179981	110	0.01602077	113	0.00893974	136	0.04098654
	5	34	0.01650405	38	0.00310564	34	0.00715566	90	0.01352477	78	0.00738406	34	0.02911496
10	1	99	0.00819159	75	0.02546334	75	0.00769567	141	0.02088284	113	0.02497864	75	0.04487300
	2	93	0.00698781	93	0.00885487	93	0.00899458	38	0.00756812	51	0.01025295	93	0.05185290
	3	113	0.01687169	36	0.01908064	114	0.00824904	55	0.01013374	51	0.00806880	36	0.04388738
	4	121	0.01907325	55	0.01899695	121	0.00902128	93	0.00838351	58	0.01727176	55	0.05068493
	5	93	0.02010012	49	0.02084494	61	0.02093768	112	0.00762677	112	0.01568270	49	0.04832864
12	1	105	0.02376676	47	0.02751851	105	0.01469874	55	0.02929068	123	0.02811289	80	0.04691601
	2	77	0.02067423	127	0.01703501	49	0.02922964	105	0.01340675	51	0.01119018	75	0.02392530
	3	127	0.02186680	119	0.01912832	58	0.02030206	100	0.02878475	68	0.02491188	68	0.03254795
	4	71	0.02586007	76	0.01875687	71	0.04331779	69	0.02734089	69	0.02265501	69	0.03074622
	5	24	0.02119994	42	0.01544404	42	0.02831125	86	0.01444960	108	0.02545428	108	0.02379966
14	1	81	0.02422452	81	0.05414796	81	0.04412675	143	0.03010392	155	0.03524613	125	0.03725243
	2	74	0.02216458	127	0.03516507	60	0.03278971	57	0.03359890	72	0.02438259	85	0.04099655
	3	133	0.02077317	105	0.03572965	97	0.02798080	127	0.03353071	62	0.03149319	67	0.03150654
	4	73	0.02979636	53	0.03472590	73	0.03740811	76	0.03250456	76	0.03215122	76	0.03961515
	5	125	0.02136493	92	0.04097843	79	0.03310132	131	0.03347325	121	0.01786423	121	0.03305411
16	1	89	0.03656673	137	0.04834127	86	0.03751588	107	0.08001876	22	0.02643538	113	0.03915858
	2	97	0.04460144	72	0.04571438	72	0.03697085	100	0.07827020	39	0.03421617	74	0.03585124
	3	74	0.03737926	161	0.04288244	151	0.04281712	45	0.08908629	59	0.03306603	59	0.06250024
	4	81	0.03455424	83	0.04537630	77	0.04227448	102	0.07774377	105	0.03344488	102	0.03752518
	5	121	0.04948711	97	0.04791141	73	0.03916359	122	0.0893116	147	0.05857158	85	0.03489161
18	1	104	0.04776144	68	0.05614901	68	0.06775546	148	0.14999485	171	0.13612795	103	0.04177213
	2	46	0.05033851	78	0.05661201	78	0.04489899	105	0.15546775	113	0.13400340	66	0.04912567
	3	72	0.03997850	63	0.05366135	97	0.03781700	100	0.14067030	145	0.13385487	65	0.06580114
	4	86	0.05907512	88	0.06118488	86	0.08527541	64	0.14927626	84	0.14838052	130	0.06495714
	5	144	0.04507351	92	0.05643296	57	0.05199432	118	0.17069912	106	0.13320255	76	0.04738760
20	1	74	0.06464696	147	0.06724644	19	0.06407595	120	0.15980697	96	0.15749955	90	0.07289219
	2	47	0.07279825	78	0.06670213	77	0.05179119	120	0.19243956	101	0.22532821	80	0.06860566
	3	109	0.07468081	129	0.07346582	76	0.05432439	125	0.16674471	52	0.24699974	45	0.05286050
	4	61	0.05321741	113	0.06722021	51	0.06233215	57	0.16668558	64	0.24061966	130	0.05959034
	5	106	0.06539440	67	0.06606865	80	0.06512165	57	0.18279243	57	0.24754643	92	0.05673695
22	1	71	0.08430076	144	0.08278370	70	0.09581637	124	0.22371101	160	0.20043278	128	0.08588076
	2	79	0.12412429	79	0.08773661	78	0.06870532	45	0.20022154	105	0.21862650	103	0.07362390
	3	105	0.07521343	150	0.09098101	74	0.07457423	90	0.21915984	151	0.22094083	98	0.07908177
	4	60	0.06681585	47	0.09236598	51	0.06897593	81	0.23369527	133	0.20637774	53	0.06499457
	5	85	0.08577561	111	0.09682608	77	0.07283950	88	0.21111965	90	0.18976593	67	0.08724236
24	1	107	0.12582541	54	0.09999275	38	0.09325886	96	0.26368928	87	0.26318669	78	0.16543555
	2	87	0.09863520	59	0.09807944	99	0.10334229	54	0.21050763	52	0.23788238	44	0.08496857
	3	124	0.09705997	156	0.10035801	46	0.08368397	137	0.26320028	11	0.23239470	53	0.09886575
	4	152	0.0918901	109	0.10085797	52	0.08409667	70	0.26605248	103	0.28364468	53	0.09179211
	5	126	0.08877373	46	0.09921670	34	0.09209132	78	0.28938794	90	0.24973154	70	0.09796929
26	1	44	0.14825583	60	0.11530089	80	0.11758876	168	0.09892654	77	0.11169004	72	0.11123443
	2	66	0.17565107	80	0.12044501	72	0.09850121	146	0.12161875	105	0.12804580	45	0.09843779
	3	66	0.18071461	93	0.11276555	64	0.11158848	83	0.10865164	92	0.09544492	71	0.10081863
	4	96	0.11376643	99	0.12015939	88	0.09683895	100	0.11655664	51	0.11103368	77	0.10978484
	5	126	0.10726023	137	0.11393547	74	0.10103488	97	0.10573244	140	0.10838437	66	0.09597945

TABLE 5. DM Variants applied to a data sample from Balas and S., 1991.

		In ascending order						In descending order					
Size	Instance	First tie-case		Last tie-case		Random tie		First tie-case		Last tie-case		Random tie	
		Cost	Runtime	Cost	Runtime	Cost	Runtime	Cost	Runtime	Cost	Runtime	Cost	Runtime
4	1	120	0.0100050	120	0.0089948	120	0.0156214	101	0.01127505	101	0.0220592	101	0.0189483
	2	64	0.0069816	64	0.0079827	64	0.0120819	51	0.0199466	51	0.0211554	51	0.01592708
	3	59	0.00797892	59	0.01592708	59	0.01972008	59	0.019909388	59	0.0209439	59	0.01396465
	4	83	0.0075715	83	0.0069814	83	0.0183339	75	0.0219560	99	0.0211179	75	0.01196718
	5	80	0.0069699	80	0.0085332	80	0.0201163	77	0.0214891	77	0.0219414	77	0.01216674
6	1	68	0.0130763	68	0.0304453	68	0.0298064	65	0.0339088	65	0.0316148	65	0.0139606
	2	115	0.0101721	115	0.0329139	115	0.0278840	63	0.0406399	63	0.0311301	63	0.0179143
	3	54	0.0115078	54	0.0289505	54	0.0279279	113	0.0369959	128	0.0324178	113	0.0228801
	4	86	0.0076144	86	0.0339091	86	0.0293858	80	0.0300484	80	0.0309000	80	0.0150967
	5	78	0.0000000	109	0.0309138	78	0.0300243	172	0.0337675	172	0.0289574	172	0.0185933
8	1	57	0.0161650	109	0.0423861	57	0.1564496	81	0.0439470	81	0.0466342	81	0.0280113
	2	122	0.0156241	77	0.0446448	68	0.1154072	49	0.0495834	87	0.0433650	49	0.0249772
	3	63	0.0211608	50	0.0439301	50	0.1348510	84	0.0384920	127	0.0434673	83	0.0317512
	4	75	0.0156887	103	0.0458648	75	0.1018567	90	0.0417278	84	0.0412576	68	0.0189126
	5	8	0.0155582	148	0.0423026	8	0.1683254	56	0.0436890	78	0.0397818	56	0.0235457
10	1	129	0.0207374	48	0.0621553	66	0.2348409	84	0.0612285	120	0.0650168	84	0.0291278
	2	74	0.0184803	74	0.0478046	74	0.2250631	52	0.0640123	53	0.0550895	52	0.0185599
	3	107	0.0168028	107	0.0557406	107	0.2193201	87	0.0725200	121	0.0549350	64	0.0249281
	4	136	0.0156217	82	0.0606716	82	0.2369609	64	0.0661914	84	0.0640216	64	0.0288825
	5	136	0.0156212	104	0.0587394	104	0.2288067	111	0.0539300	113	0.0668213	111	0.0238676
12	1	68	0.0249102	66	0.0990796	64	0.0758309	75	0.0793335	57	0.0907576	57	0.0391731
	2	125	0.0230906	122	0.0932608	84	0.0831053	128	0.0953901	77	0.0924416	65	0.0339096
	3	98	0.0162509	115	0.0947340	79	0.0843966	112	0.0943108	112	0.0961335	112	0.0218928
	4	69	0.0156663	90	0.0980532	69	0.0822625	81	0.0933189	81	0.0928619	81	0.0319502
	5	174	0.0312402	76	0.0936139	76	0.0833957	116	0.0904186	101	0.0932014	101	0.0398910
14	1	66	0.0319147	141	0.1309872	35	0.1052270	148	0.1092475	69	0.1226533	69	0.1057429
	2	24	0.0275707	114	0.1384912	53	0.0963893	125	0.1075077	113	0.1115911	31	0.1259358
	3	111	0.0200613	93	0.1388362	77	0.0976758	104	0.1226351	53	0.1174052	53	0.1195037
	4	118	0.0361946	103	0.1445396	50	0.1257784	61	0.1180797	40	0.1215775	40	0.0935991
	5	43	0.0324488	172	0.1199877	38	0.1163676	50	0.1126332	96	0.1221256	38	0.0962217
16	1	59	0.0388928	130	0.1778674	52	0.1228278	107	0.1657932	60	0.1739564	40	0.1268768
	2	92	0.0388606	119	0.2036905	46	0.1422348	129	0.1772234	86	0.1684265	48	0.1388183
	3	90	0.0389342	125	0.1923089	71	0.1347578	23	0.1709454	149	0.1818943	70	0.1187835
	4	95	0.0390573	91	0.2071362	74	0.1140461	135	0.1625075	70	0.1704474	63	0.1274412
	5	69	0.0397661	96	0.1989081	70	0.1212702	34	0.1775985	61	0.1776381	35	0.1340263
18	1	51	0.0450637	87	0.2627711	43	0.1515684	124	0.2528760	136	0.2504299	50	0.1642561
	2	49	0.0458317	71	0.2908456	38	0.1409218	110	0.2543094	70	0.2725921	31	0.1618323
	3	142	0.0527828	37	0.2641921	52	0.1567829	114	0.2464047	139	0.2352862	33	0.1620760
	4	119	0.0489056	130	0.2554598	78	0.1542656	82	0.2425082	80	0.2451539	55	0.1572001
	5	47	0.0526750	84	0.2798693	73	0.1613507	104	0.2343776	114	0.2317057	61	0.1522055
20	1	105	0.0627451	94	0.3496108	47	0.0634198	64	0.3394494	40	0.3405883	26	0.1952128
	2	30	0.0549486	82	0.3579905	34	0.0693061	119	0.3286281	49	0.3341446	49	0.2078474
	3	122	0.0533302	117	0.3564801	65	0.0708215	94	0.3489153	126	0.3453660	67	0.2066286
	4	53	0.0653865	88	0.3614526	60	0.0738537	91	0.3345735	115	0.3355937	56	0.1855583
	5	122	0.0502832	90	0.3462441	30	0.0708125	80	0.3105972	76	0.3217070	37	0.1937101
22	1	47	0.0782130	164	0.4508257	47	0.0881426	154	0.4368594	140	0.4384162	51	0.2324278
	2	88	0.0697444	175	0.4765818	33	0.0754585	112	0.4382422	102	0.4250071	37	0.2334025
	3	75	0.0625374	125	0.4731760	37	0.0791652	108	0.4618361	73	0.4435165	49	0.2347534
	4	94	0.0621159	106	0.4753542	71	0.0879195	113	0.4583137	96	0.4565649	47	0.2089188
	5	113	0.0863214	46	0.4623990	52	0.0848382	69	0.4483247	116	0.4604309	52	0.2614031
24	1	77	0.0803669	137	0.6145887	41	0.0959871	169	0.5765002	128	0.5605409	59	0.2462826
	2	135	0.0945604	145	0.6040702	53	0.0909970	94	0.5608380	159	0.6228988	23	0.2617285
	3	65	0.0957706	136	0.6173430	33	0.0960672	102	0.5701540	78	0.6006057	39	0.2665687
	4	105	0.0892520	105	0.5927818	51	0.0954394	92	0.5686386	132	0.5602443	45	0.2553465
	5	113	0.0896075	90	0.5766120	38	0.0963628	139	0.5742068	94	0.6153796	31	0.2952910
26	1	66	0.1067441	96	0.7707765	46	0.1065700	109	0.7439399	69	0.7319100	39	0.3155532
	2	77	0.0983267	64	0.7752740	40	0.1063283	87	0.7136793	97	0.7285006	31	0.3208118
	3	88	0.1091614	107	0.7605805	44	0.1111016	112	0.7399192	61	0.7150211	44	0.2938275
	4	100	0.0911787	85	0.7670391	49	0.1214638	158	0.7498312	59	0.7476666	48	0.3117094
	5	63	0.1003244	84	0.7637739	37	0.1106565	120	0.7235053	50	0.7177529	43	0.3406253

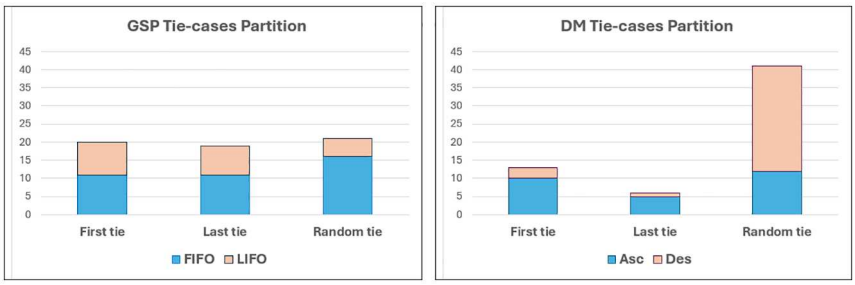

FIGURE 2. The partition of the best solutions found per variant.

The heuristics introduced in this study yield multiple solutions for 3IAP, whereas the original version of DM generates merely one or two, focusing solely on the first tie-cases. Our examination indicates that the larger the set of solutions, the better the quality of the obtained solution. A comparative analysis of the DM results for the first tie-cases in Table 5 against the combined GSP and DM variants outcomes in the last column of Table 6 confirms the primacy of our heuristics for addressing 3IAP, underscoring their efficacy.

Computational tests suggest that developing new variants of GSP and DM enables achieving feasible solutions, which often outperform those derived from the DM simple version, as displayed in Figure 3.

DM operates through two phases; the first orders the cost matrices, while the second, GSP, allows us to attain a feasible solution for 3IAP. The results in Table 6 indicate that the GSP should be considered because it could lead to better solutions. In the following, we discuss two heuristic classes designed for swift, high-quality approximate solutions for 3IAP, substantiated by extensive numerical testing. The first category relies on statistical measures of the cost matrices, whereas the second on matrix norms.

5.4. Heuristics of category 1

The rationale for sorting the cost matrices according to their diagonal factors is omitted in [25], a method not previously used or mentioned. The current research investigates alternate matrices sorting methodologies introducing two novel heuristic categories aligned with the DM framework. The first category of heuristics employs descriptive statistical measures, while the second refers to matrix norms. This inquiry not only questions the singular use of the diagonals method but also broadens the scope of strategies for matrix arrangement in optimization problems.

The objective function of 3IAP is defined by n positive matrices of $\mathcal{M}^{n^2}$, each associated with a factory. Each cost matrix is treated as one-dimensional vector of $\mathbb{R}^{n^2}$. Consequently, vector norms serve as criteria for sorting the cost matrices, facilitating a systematic method for their organization.

Among vector norms defined over the vector space $\mathbb{R}^n$, the 1-norm and the infinity-norm are considered in the following. Let C be a cost matrix, associated with an arbitrary factory. Let us consider matrix C as a vector of $\mathbb{R}^{n^2}$.

Given that all coefficients of the matrix costs are nonnegative, the 1-norm of C is $\| C \|_1 = \sum_{i=1}^{n^2} c_i$ and if divided by n^2, it equates to $mean(C)$. This implies that ranking cost matrices by their mean values is equivalent to sorting them by their 1-norms. Therefore, 3IAP can be effectively addressed by reordering its matrices based on their mean values in ascending or descending order, supporting a streamlined solution approach.

However, if the costs take extreme values, it is advisable to sort the matrices based on their median rather than their mean to prevent their order from being affected by the outliers. Rearranging the matrices according to their median in either increasing or decreasing order, then applying GSP to obtain a feasible solution to 3IAP that may yield a different solution.

The infinity-norm, $\| C \|_\infty$ equals the maximum coefficient of C since all costs are nonnegative. The matrices here are treated as vectors and can be rearranged according to their maximum; subsequently they can be

TABLE 6. Summary of the results of GSP and DM methods.

		GSP heuristic				DM method				Best solution	
Size	Instance	Cost	Runtime	Tie Rule		Cost	Runtime	Tie Order		Cost	Runtime
4	1	46	0.00210214	First	Fifo	101	0.01127505	First	Desc	46	0.00210214
	2	73	0.00338864	Last	Lifo	51	0.01592710	Rand	Desc	51	0.01592710
	3	54	0.00194740	Last	Fifo	59	0.01972008	Last	Asc	54	0.00194740
	4	75	0.00362301	First	Lifo	75	0.01196718	Rand	Desc	75	0.00362301
	5	50	0.00099754	First	Fifo	77	0.01216674	Rand	Desc	50	0.00099754
6	1	73	0.00567222	First	Fifo	65	0.0139606	Rand	Desc	65	0.0139606
	2	51	0.00423217	First	Fifo	63	0.01791430	Rand	Desc	51	0.00423217
	3	101	0.00940967	First	Fifo	54	0.01150788	First	Asc	54	0.01150788
	4	113	0.00099707	Last	Fifo	80	0.01509666	Rand	Desc	80	0.01509666
	5	73	0.00000000	Last	Fifo	78	0.00000000	First	Asc	73	0.00000000
8	1	45	0.00316525	First	Fifo	57	0.01616502	First	Asc	45	0.00316525
	2	66	0.00710893	Last	Lifo	49	0.02497721	Rand	Desc	49	0.02497712
	3	46	0.04327226	Rand	Lifo	50	0.04393005	Last	Asc	46	0.04327226
	4	110	0.01602077	First	Lifo	68	0.01891255	Rand	Desc	68	0.01891255
	5	34	0.00715566	Rand	Fifo	8	0.01555824	First	Asc	8	0.01555824
10	1	75	0.00769567	Rand	Fifo	48	0.06215525	Last	Asc	48	0.06215525
	2	38	0.00756812	First	Lifo	52	0.01855993	Rand	Desc	38	0.00756812
	3	36	0.01908064	Last	Fifo	64	0.02492809	Rand	Desc	36	0.01908064
	4	55	0.01899695	Last	Fifo	64	0.02888250	Rand	Desc	55	0.01899695
	5	49	0.02084494	Last	Fifo	104	0.05873942	Last	Asc	49	0.02084494
12	1	47	0.02751851	Last	Fifo	57	0.03917313	Rand	Desc	47	0.02751851
	2	49	0.02922964	Rand	Fifo	65	0.03390956	Rand	Desc	49	0.02922964
	3	58	0.02030206	Rand	Fifo	79	0.08439660	Rand	Asc	58	0.02030206
	4	69	0.02265501	Last	Lifo	69	0.0156663	First	Asc	69	0.01566625
	5	24	0.02119994	First	Fifo	76	0.08339572	Rand	Asc	24	0.02119994
14	1	81	0.02422452	First	Fifo	35	0.10522699	Rand	Asc	35	0.10522699
	2	57	0.03359890	First	Lifo	24	0.02757072	First	Asc	24	0.02757072
	3	62	0.03149319	Last	Lifo	53	0.11740518	Last	Desc	53	0.11740518
	4	53	0.03472590	Last	Fifo	40	0.09359908	Rand	Desc	40	0.09359908
	5	79	0.03310132	Rand	Fifo	38	0.09622169	Rand	Desc	38	0.09622169
16	1	22	0.02643538	Last	Lifo	40	0.12687683	Rand	Desc	22	0.02643538
	2	39	0.03421617	Last	Lifo	46	0.14223480	Rand	Asc	39	0.03421617
	3	45	0.08908629	First	Lifo	23	0.17094541	First	Desc	23	0.17094541
	4	77	0.04227448	Rand	Fifo	63	0.12744117	Rand	Desc	63	0.12744117
	5	73	0.03916359	Rand	Fifo	34	0.17759848	First	Desc	34	0.17759848
18	1	68	0.05614901	Last	Fifo	43	0.15156841	Rand	Asc	43	0.15156841
	2	46	0.05033851	First	Fifo	31	0.16183233	Rand	Desc	31	0.16183233
	3	63	0.05366135	Last	Fifo	33	0.16207600	Rand	Desc	33	0.16207600
	4	64	0.14927626	First	lifo	55	0.15720010	Rand	Desc	55	0.15720010
	5	57	0.05199432	Rand	Fifo	47	0.05267501	First	Asc	47	0.05267501
20	1	19	0.06407595	Rand	Fifo	26	0.19521284	Rand	Desc	19	0.06407595
	2	47	0.07279825	First	Fifo	30	0.68915153	First	Asc	30	0.68915153
	3	45	0.05286050	First	Lifo	65	0.07082152	Rand	Asc	45	0.05286050
	4	51	0.06233215	Rand	Fifo	53	0.65729141	First	Asc	51	0.06233215
	5	57	0.18279243	First	Lifo	30	0.07081246	Rand	Asc	30	0.07081246
22	1	70	0.09581637	Rand	Fifo	47	0.07821298	First	Asc	47	0.07821298
	2	45	0.20022154	First	Lifo	33	0.07545853	Rand	Asc	33	0.07545853
	3	74	0.07457423	Rand	Fifo	37	0.07916522	Rand	Asc	37	0.07916522
	4	47	0.09236598	Last	Fifo	47	0.20891881	Rand	Desc	47	0.09236598
	5	67	0.08724236	Rand	Lifo	46	0.46239901	Last	Asc	46	0.46239901
24	1	38	0.09325886	Rand	Fifo	41	0.09598708	Rand	Asc	38	0.09325886
	2	44	0.08496857	Rand	Lifo	23	0.26172853	Rand	Desc	23	0.26172853
	3	11	0.23239470	Last	Lifo	33	0.09606719	Rand	Asc	11	0.23239470
	4	52	0.08409667	Rand	Fifo	45	0.25534654	Rand	Desc	45	0.25534654
	5	34	0.09209132	Rand	Fifo	31	0.29529095	Rand	Desc	31	0.29529095
26	1	44	0.14825583	First	Fifo	39	0.31555319	Rand	Desc	39	0.31555319
	2	45	0.09843779	Rand	Lifo	31	0.32081175	Rand	Desc	31	0.32081175
	3	64	0.11158848	Rand	Fifo	44	0.11110163	Rand	Desc	44	0.11110163
	4	51	0.11103368	Last	Lifo	48	0.31170940	Rand	Desc	48	0.31170940
	5	66	0.09597945	Rand	Lifo	37	0.11065650	Rand	Asc	37	0.11065650

sorted according to their coefficients range. Consequently 3IAP will have two solutions, one from sorting its cost matrices according to their maxima and the second achieved after rearranging the matrices according to their range.

Similarly, the standard deviation, a common measure of dispersion in descriptive statistics, can be considered. Reorganizing the matrices based on their standard deviation, in ascending or descending order, and then applying GSP leads to feasible solutions for 3IAP.

In summary, the development of five novel heuristics in addition to the GSP and DM, through sorting cost matrices by their mean, median, maximum, range, or standard deviation, forms a new category designated as

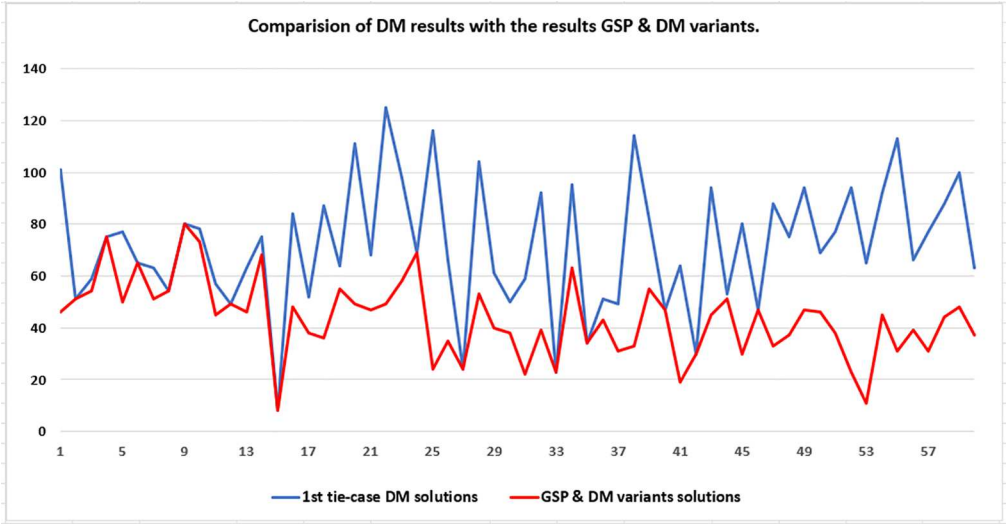

FIGURE 3. Comparison of DM results with the results of GSP and DM variants.

TABLE 7. Solutions for a 3IAP instance of size 10, selecting the first tie-case.

Index	Sort type	Order	Cost	Runtime
1	GSP	–	133	0.01562381
2	Diagonals	Ascending	107	0.01562095
3	Diagonals	Descending	87	0.01701713
4	Maximum	Ascending	160	0.01566124
5	Maximum	Descending	84	0.03124285
6	Mean	Ascending	71	0.01836586
7	Mean	Descending	58	0.01920557
8	Median	Ascending	165	0.02127552
9	Median	Descending	123	0.02434134
10	Range	Ascending	72	0.02418661
11	Range	Descending	86	0.02154136
12	Std. Deviation	Ascending	88	0.01696014
13	Std. Deviation	Descending	92	0.01572490

Category 1. These heuristics predicated on statistical sorting criteria, account for ten methodologies in both ascending and descending orders. Alongside a solution derived from GSP applied to unsorted matrices, this approach rapidly yields eleven additional feasible solutions for each instance of 3IAP. The best found solution is then selected from these alternatives.

To apply Category 1 heuristics to 3IAP, let us choose from Sample 1, for example, an instance 3 of size 10. The problem handles 1,000 variables with 1.3×10^{13} potential feasible solutions. The heuristics operate concurrently across the first, last, and random tie-cases, resulting in 13 outcomes for each case. The solutions yielded are presented in Tables 7, 8, and 9, respectively. The best solution found in each table is underlined.

Table 10 summarises the experimental results obtained from these three tables.

Next, for each instance these heuristics are run six times, one for each variant of the tie-cases, multiplied by two for ascending and descending sorting orders. The program will retain only the best solution obtained for each instance.

TABLE 8. Solutions for a 3IAP instance of size 10, selecting the last tie-case.

Index	Sort type	Order	Cost	Runtime
1	GSP	–	36	0.03793907
2	Diagonals	Ascending	107	0.03272176
3	Diagonals	Descending	121	0.02891898
4	Maximum	Ascending	160	0.02564788
5	Maximum	Descending	75	0.02691746
6	Mean	Ascending	104	0.02027655
7	Mean	Descending	58	0.02844763
8	Median	Ascending	165	0.02450919
9	Median	Descending	123	0.02194047
10	Range	Ascending	72	0.00607562
11	Range	Descending	69	0.02133369
12	Std. Deviation	Ascending	108	0.02435017
13	Std. Deviation	Descending	130	0.03312850

TABLE 9. Solutions for a 3IAP instance of size 10, selecting a tie-case at random.

Index	Sort type	Order	Cost	Runtime
1	GSP	–	133	0.02292633
2	Diagonals	Ascending	107	0.00855851
3	Diagonals	Descending	27	0.03613329
4	Maximum	Ascending	160	0.00891209
5	Maximum	Descending	75	0.03389192
6	Mean	Ascending	104	0.01489067
7	Mean	Descending	58	0.01526475
8	Median	Ascending	165	0.03229070
9	Median	Descending	123	0.02351594
10	Range	Ascending	72	0.02376366
11	Range	Descending	72	0.02641940
12	Std. Deviation	Ascending	79	0.02238274
13	Std. Deviation	Descending	126	0.02317810

TABLE 10. Summary of best solutions for a 3IAP instance of size 10.

Case	Sort type	Order	Cost	Runtime
First tie-case	Mean	Descending	58	0.01920557
Last tie-case	GSP	–	36	0.03793907
Random tie	Diagonals	Descending	27	0.03613329

To evaluate the efficiency of Category 1 heuristics, additional tests were conducted on Sample 1 data, with the findings summarized in Table 11. Each row represents an instance, highlighting the best solution achieved. It is observed that of the 60 best solutions identified, only six were yielded by the original DM version *i.e.*, 10%, while the remaining were generated by Category 1 heuristics.

To summarize, Category 1 heuristics not only provide feasible solutions for 3IAP but they also outperform the DM algorithm. Among these solutions, some are optimal and others serve as effective starting points for

TABLE 11. Outcomes of Category 1 heuristics implemented on Sample 1.

		First tie-case				Last tie-case				Random tie-case			
Size	Instance	Type	Order	Cost	Runtime	Type	Order	Cost	Runtime	Type	Order	Cost	Runtime
	1	Std. Dev.	Asc	46	0.0153406	Unsorted	–	46	0.0159895	Maximum	Desc	46	0.0079737
	2	Diagonals	Desc	51	0.0197680	Diagonals	Desc	51	0.0256345	Diagonals	Desc	51	0.0079789
4	3	Std. Dev.	Desc	54	0.0172117	Unsorted	Desc	54	0.0160148	Unsorted	—	54	0.0080109
	4	Mean	Asc	68	0.0279086	Mean	Asc	71	0.0256448	Mean	Asc	68	0.0089753
	5	Mean	Desc	42	0.0209127	Std. Dev.	Desc	22	0.0269482	Std. Dev.	Desc	22	0.0089760
	1	Std. Dev.	Asc	54	0.0382040	Maximum	Asc	60	0.0436385	Median	Desc	42	0.0119770
	2	Unsorted	—	51	0.0320008	Unsorted	—	51	0.0334609	Unsorted	—	51	0.0119681
6	3	Median	Desc	42	0.0349155	Median	Desc	42	0.0320060	Median	Desc	42	0.0119572
	4	Mean	Desc	50	0.0255313	Std. Dev.	Desc	50	0.0348604	Median	Desc	50	0.0119636
	5	Range	Desc	28	0.0404816	Range	Desc	28	0.0426095	Range	Desc	28	0.0119267
	1	Range	Desc	37	0.0416465	Median	Desc	36	0.0465174	Median	Desc	36	0.0510972
	2	Maximum	Desc	44	0.0426154	Range	Asc	55	0.0400841	Maximum	Desc	44	0.0510507
8	3	Std. Dev.	Asc	35	0.0463522	Std. Dev.	Asc	35	0.0385490	Std. Dev.	Asc	35	0.0169549
	4	Median	Desc	57	0.0424366	Median	Asc	56	0.0568943	Median	Asc	56	0.0534122
	5	Diagonals	Asc	8	0.0399439	Range	Desc	8	0.0400512	Range	Desc	8	0.0486159
	1	Maximum	Desc	37	0.0156262	Maximum	Desc	37	0.0561564	Maximum	Desc	37	0.0630534
	2	Diagonals	Desc	52	0.0545759	Range	Asc	43	0.0735247	Range	Asc	43	0.0711098
10	3	Mean	Des	58	0.01920557	Unsorted	—	36	0.03793907	Diagonals	Desc	27	0.03613329
	4	Median	Desc	32	0.0554729	Median	Desc	32	0.0626526	Range	Desc	32	0.0647616
	5	Maximum	Desc	43	0.0587294	Median	Asc	35	0.0667386	Mean	Asc	40	0.0721371
	1	Median	Asc	62	0.0872922	Range	Asc	39	0.0839060	Range	Asc	75	0.0875769
	2	Range	Asc	65	0.0816712	Std. Dev.	Asc	53	0.0730062	Std. Dev.	Asc	53	0.0845430
12	3	Mean	Desc	36	0.0703671	Range	Desc	61	0.0726814	Mean	Desc	36	0.0831275
	4	Range	Desc	64	0.0884013	Std Dev.	Asc	63	0.0789812	Range	Desc	64	0.0755398
	5	Unsorted	—	24	0.0794580	Unsorted	—	42	0.0701621	Mean	Desc	37	0.0860555
	1	Mean	Asc	33	0.1059499	Median	Desc	49	0.1009805	Maximum	Desc	49	0.1198707
	2	Diagonals	Asc	24	0.1002367	Range	Asc	41	0.1020365	Maximum	Asc	55	0.1281445
14	3	Range	Asc	28	0.0981150	Mean	Desc	31	0.1226015	Median	Desc	36	0.1004131
	4	Std. Dev.	Desc	44	0.0881739	Diagonals	Desc	40	0.1018407	Diagonals	Desc	40	0.1043491
	5	Diagonals	Asc	43	0.0979013	Mean	Asc	68	0.1243742	Median	Desc	62	0.1281118
	1	Median	Desc	47	0.1142566	Maximum	Asc	60	0.1084011	Median	Desc	47	0.1223409
	2	Range	Asc	64	0.0952747	Std. Dev.	Desc	52	0.1191566	Range	Desc	28	0.1472983
16	3	Diagonals	Desc	23	0.1209276	Mean	Desc	43	0.1139445	Median	Desc	42	0.1242037
	4	Mean	Asc	52	0.1106885	Maximum	Desc	70	0.1087453	Median	Desc	32	0.1276939
	5	Diagonals	Desc	34	0.0470531	Median	Desc	41	0.1305757	Mean	Asc	43	0.1306794
	1	Maximum	Asc	25	0.0548527	Std. Dev.	Asc	40	0.1483743	Range	Desc	36	0.1832347
	2	Std. Dev.	Asc	37	0.0568481	Median	Desc	52	0.1427739	Median	Desc	32	0.1759551
18	3	Range	Desc	55	0.0588415	Diagonals	Asc	37	0.1669588	Std. Dev.	Desc	69	0.1584294
	4	Mean	Asc	43	0.0628331	Std Dev.	Asc	51	0.1522281	Range	Desc	46	0.1755247
	5	Range	Desc	31	0.0528653	Mean	Asc	54	0.1569455	Range	Desc	31	0.1566219
	1	Std. Dev.	Desc	53	0.0698125	Diagonals	Desc	40	0.1883280	Maximum	Desc	52	0.2294993
	2	Maximum	Desc	29	0.0802653	Diagonals	Desc	49	0.1882982	Std. Dev.	Asc	64	0.2192030
20	3	Median	Desc	58	0.0658586	Median	Desc	77	0.1917918	Mean	Asc	58	0.1976802
	4	Diagonals	Asc	53	0.0703182	Range	Desc	66	0.1934402	Diagonals	Asc	47	0.2138100
	5	Range	Desc	25	0.0678186	Maximum	Desc	58	0.1991711	Range	Asc	39	0.2129843
	1	Mean	Asc	42	0.0928795	Maximum	Asc	30	0.2470129	Maximum	Asc	44	0.0967760
	2	Maximum	Asc	53	0.0837317	Mean	Desc	41	0.2319818	Std. Dev.	Asc	65	0.2580817
22	3	Median	Desc	39	0.0877655	Range	Asc	54	0.2341025	Range	Desc	55	0.2452874
	4	Mean	Desc	43	0.0837760	Mean	Desc	26	0.2202597	Mean	Asc	53	0.0917525
	5	Mean	Desc	47	0.0867689	Std. Dev.	Asc	39	0.2596839	Std. Dev.	Asc	57	0.0907536
	1	Median	Asc	26	0.1042540	Mean	Asc	49	0.2847967	Std. Dev.	Asc	27	0.1067152
	2	Median	Desc	55	0.0987363	Median	Asc	40	0.2738364	Range	Asc	58	0.1057088
24	3	Range	Desc	52	0.0987296	Std. Dev.	Asc	34	0.2768462	Std. Dev.	Desc	57	0.1056824
	4	Median	Desc	72	0.1017270	Mean	Asc	62	0.2795932	Std. Dev.	Desc	27	0.1047204
	5	Std Dev.	Desc	45	0.1062431	Mean	Desc	41	0.2797968	Unsorted	Desc	47	0.1082633
	1	Unsorted	—	44	0.1361423	Unsorted	—	60	0.3399053	Mean	Asc	39	0.1266959
	2	Median	Desc	32	0.1186829	Maximum	Asc	54	0.3262534	Median	Asc	64	0.1286516
26	3	Std. Dev.	Asc	62	0.1196902	Maximum	Asc	30	0.3403540	Median	Asc	52	0.1311564
	4	Diagonals	Desc	43	0.1216748	Diagonals	Desc	59	0.3425541	Mean	Asc	68	0.1276686
	5	Diagonals	Asc	63	0.1216748	Diagonals	Desc	50	0.3457615	Std. Dev.	Desc	33	0.1206770

TABLE 12. Solutions for a 3IAP instance of size 10, selecting the first tie-case.

Index	Sort type	Order	Cost	Runtime
1	Unsorted	–	133	0.00423265
2	Diagonals	Ascending	107	0.01365447
3	Diagonals	Descending	87	0.00911498
4	Frobenius	Ascending	71	0.01427603
5	Frobenius	Descending	58	0.01166511
6	1-norm	Ascending	83	0.01676869
7	1-norm	Descending	56	0.00988197
8	Infinity	Ascending	83	0.02835202
9	Infinity	Descending	69	0.02092838
10	2-norm	Ascending	122	0.01850247
11	2-norm	Descending	55	0.00955343

the B&B method. Typically, the hybrid heuristics of Category 1 are associated with producing high-quality solutions. In the following section, another category of heuristics for 3IAP will be examined.

5.5. Heuristics of category 2

The study proposes a new criterion for reorganizing cost matrices by employing matrix norms instead of statistical measures. This approach leads to the design of four novel heuristics, termed Category 2, analogous to the heuristics of Category 1. The methodology promises enhanced efficacy in 3IAP solution approaches.

The 3IAP objective function is defined by cost matrices, which are two-dimensional arrays where each coefficient value represents the cost associated with a specific row-column assignment. These matrices are elements of $\mathcal{M}^{n^2}$; thus, to rearrange them, it is helpful to refer to matrix norms. There are four common norms defined over the vector space $\mathcal{M}^{n^2}$.

Let A be an element of $\mathcal{M}^{n^2}$.

(1) **Frobenius norm**: $\| A \|_F$ is equal to the square root of the sum of all the squares of the coefficients of A. $\| A \|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^2}$.
(2) **1-norm**: $\| A \|_1$ is equal to the maximum of the sum of the absolute values of the coefficients of the columns of A. $\| A \|_1 = \max_{\|x\|=1} \| Ax \|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} | a_{ij} |$.
(3) **Infinity norm**: $\| A \|_\infty$ is the maximum of the sum of the absolute values of the coefficients of the rows of A. $\| A \|_\infty = \max_{\|x\|=1} \| Ax \|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} | a_{ij} |$.
(4) **2-norm**: $\| A \|_2$ is the square root of the dominant eigenvalue of the inner product of $A^T A$. $\| A \|_2 = \max_{\|x\|=1} \| Ax \|_2 = \max_{1 \le i \le n} \sqrt{\lambda_i (A^T A)}$.

For comparative analysis, Category 2 heuristics are implemented on the same 3IAP instance of size 10, operating across the first-tie, last-tie, and random tie-cases. This implementation results in eleven solutions per case, documented in Tables 12, 13, and 14, respectively. In each row the best solution found is highlighted by underlining, showcasing the efficiency of the heuristics.

Tables 12, 13, and 14, display eight solutions from Category 2 heuristics for an instance of size 10, compared with GSP and DM results where the best solutions are highlighted. Category 2 heuristics outperform the basic DM version, showing improvements in objective function values and computational efficiency, illustrating the effectiveness of these heuristics. Table 15 summarises the best solutions achieved within the three previous tables.

Table 16 presents the results from applying Category 2 heuristics to Sample 1, with computational tests mirroring those in Table 11. The best-found solution for each 3IAP instance is underlined. Out of the 60

TABLE 13. Solutions for a 3IAP instance of size 10, selecting the last tie-case.

Index	Sort type	Order	Cost	Runtime
1	Unsorted	–	36	0.01769304
2	Diagonals	Ascending	107	0.02649331
3	Diagonals	Descending	121	0.03093147
4	Frobenius	Ascending	104	0.02595663
5	Frobenius	Descending	58	0.01140642
6	1-norm	Ascending	90	0.02021837
7	1-norm	Descending	56	0.01440573
8	Infinity	Ascending	27	0.02520680
9	Infinity	Descending	69	0.01849794
10	2-norm	Ascending	92	0.01979470
11	2-norm	Descending	54	0.01054478

TABLE 14. Solutions for a 3IAP instance of size 10, selecting a random tie.

Index	Sort type	Order	Cost	Runtime
1	Unsorted	–	133	0.01906562
2	Diagonals	Ascending	107	0.02649331
3	Diagonals	Descending	121	0.03093147
4	Frobenius	Ascending	104	0.01951051
5	Frobenius	Descending	58	0.01783252
6	1-norm	Ascending	166	0.02023554
7	1-norm	Descending	56	0.00896358
8	Infinity	Ascending	27	0.02271700
9	Infinity	Descending	69	0.02086163
10	2-norm	Ascending	122	0.02661896
11	2-norm	Descending	54	0.02083349

TABLE 15. Summary of best solutions for a 3IAP instance of size 10.

Case	Sort type	Order	Cost	Runtime
First tie-case	2-Norm	Descending	55	0.00955343
Last tie-case	Infinity	Ascending	27	0.02520680
Random tie	Infinity	Ascending	27	0.02271700

premier solutions identified, only eleven are the outcome of DM, with the remainder arising from Category 2 heuristics. Combining data from Tables 11 and 16 yields solutions superior to those achieved through DM alone. Furthermore, even when cost values are identical for specific instances, solutions obtained by these heuristics feature reduced computational time.

Table 17 delineates the newly proposed heuristics for 3IAP, and the rearranged cost matrices according to two criteria: statistical measures and matrix norms. The approach generates two sets of feasible solutions for 3IAP and in addition to those derived from DM and GSP methods. As a result, each 3IAP instance is provided with 21 feasible solutions within a competitive timeframe, ensuring a commendable solution for the problem.

TABLE 16. Outcomes of Category 2 heuristics implemented on Sample 1.

Size	Instance	First tie-case				Last tie-case				Random tie-case			
		Type	Order	Cost	Runtime	Type	Order	Cost	Runtime	Type	Order	Cost	Runtime
4	1	Unsorted	–	46	0.0169406	Unsorted	–	46	0.0010352	Unsorted	–	46	0.0010238
	2	Diagonals	Desc	51	0.0202684	Diagonals	Desc	51	0.0043216	Diagonals	Desc	51	0.0033302
	3	Unsorted	–	54	0.0219405	2-Norm	Desc	54	0.0010223	Unsorted	–	54	0.0069017
	4	Frobenius	Asc	68	0.0237219	Frobenius	Asc	71	0.0075302	2-Norm	Asc	71	0.0081911
	5	1-Norm	Asc	40	0.0222788	Infinity	Asc	22	0.0080113	Frobenius	Desc	22	0.0070922
6	1	Diagonals	Desc	65	0.0166636	Diagonals	Desc	65	0.0103643	Diagonals	Desc	65	0.0160758
	2	Frobenius	Asc	42	0.0190020	Frobenius	Asc	42	0.0117474	Frobenius	Asc	42	0.0150635
	3	Frobenius	Desc	48	0.0179522	Frobenius	Desc	48	0.0123291	Frobenius	Desc	48	0.0146244
	4	2-Norm	Desc	50	0.0378447	2-Norm	Desc	50	0.0109694	2-Norm	Desc	50	0.0146244
	5	Infinity	Asc	45	0.0270686	Infinity	Asc	45	0.0120251	Infinity	Asc	45	0.0159576
8	1	1-Norm	Desc	37	0.0246043	1-Norm	Desc	37	0.0139608	1-Norm	Desc	37	0.0010242
	2	1-Norm	Desc	49	0.0176947	1-Norm	Desc	49	0.0136608	1-Norm	Desc	49	0.0036578
	3	Infinity	Asc	35	0.02398442	1-Norm	Desc	47	0.0139973	Unsorted	–	46	0.0143979
	4	Frobenius	Desc	57	0.0392866	2-Norm	Asc	56	0.0042777	2-Norm	Asc	56	0.0127425
	5	1-Norm	Desc	8	0.0116396	1-Norm	Desc	8	0.0035248	1-Norm	Desc	8	0.0146234
10	1	2-Norm	Asc	71	0.0622561	Diagonals	Asc	48	0.0169549	2-Norm	Asc	71	0.0219409
	2	Infinity	Desc	40	0.0678885	2-Norm	Desc	47	0.0166063	Diagonals	Desc	53	0.0188389
	3	2-Norm	Desc	55	0.00955343	Infinity	Asc	27	0.02520680	Infinity	Asc	27	0.02271700
	4	2-Norm	Desc	44	0.0675690	2-Norm	Desc	44	0.0169876	2-Norm	Desc	44	0.0165770
	5	Infinity	Desc	79	0.0411978	Unsorted	–	49	0.0179517	2-Norm	Asc	43	0.0197444
12	1	1-Norm	Desc	59	0.0701144	Unsorted	–	47	0.0200324	1-Norm	Desc	59	0.0356193
	2	Unsorted	–	77	0.0310483	Infinity	Desc	29	0.0189824	Unsorted	–	59	0.0249324
	3	Infinity	Desc	60	0.0712464	Frobenius	Asc	28	0.0194366	Infinity	Desc	51	0.0110812
	4	1-Norm	Desc	40	0.0324903	Infinity	Asc	56	0.0233817	Frobenius	Asc	48	0.0222557
	5	Unsorted	–	24	0.0301309	Unsorted	–	42	0.0203908	2-Norm	Desc	37	0.0255072
14	1	Diagonals	Asc	66	0.0860665	Frobenius	Desc	46	0.0239670	Diagonals	Asc	35	0.0315444
	2	Diagonals	Asc	24	0.0957935	1-Norm	Desc	40	0.0209870	2-Norm	Desc	39	0.0297997
	3	1-Norm	Asc	90	0.0504882	2-Norm	Desc	42	0.0094311	1-Norm	Desc	38	0.0313828
	4	Diagonals	Desc	61	0.0547142	Diagonals	Desc	40	0.0110421	Unsorted	–	56	0.0350776
	5	Infinity	Asc	29	0.0520158	2-Norm	Desc	48	0.0216281	Diagonals	Desc	45	0.0413561
16	1	Infinity	Desc	34	0.064518	Diagonals	Desc	60	0.0292914	Diagonals	Desc	40	0.0228353
	2	Infinity	Desc	42	0.0678213	Infinity	Desc	34	0.0294390	Infinity	Desc	42	0.0267251
	3	Diagonals	Desc	23	0.0832505	1-Norm	Desc	42	0.0248818	Frobenius	Asc	68	0.0322201
	4	1-Norm	Desc	69	0.0899043	Frobenius	Desc	66	0.0230651	Infinity	Desc	48	0.0317247
	5	Diagonals	Desc	34	0.0840192	Diagonals	Desc	61	0.0298672	Infinity	Asc	43	0.0348649
18	1	Diagonals	Asc	51	0.0941238	Unsorted	–	68	0.0190866	2-Norm	Asc	47	0.0378969
	2	2-Norm	Desc	30	0.0986674	Frobenius	Desc	35	0.0347402	Unsorted	–	46	0.0349045
	3	Frobenius	Desc	46	0.1041887	Diagonals	Asc	37	0.0251563	Diagonals	Desc	33	0.0418868
	4	1-Norm	Desc	56	0.0944135	1-Norm	Desc	62	0.0309236	2-Norm	Asc	47	0.0388122
	5	Diagonals	Asc	47	0.0936954	Infinity	Desc	42	0.0311666	Diagonals	Desc	53	0.0366027
20	1	Frobenius	Desc	18	0.0948539	Infinity	Desc	36	0.0369012	2-Norm	Desc	26	0.0404551
	2	Diagonals	Asc	30	0.0942195	Frobenius	Asc	33	0.0341244	Diagonals	Asc	45	0.0524790
	3	2-Norm	Asc	57	0.1218150	2-Norm	Desc	62	0.0340006	2-Norm	Asc	58	0.0468812
	4	Infinity	Asc	38	0.1433766	2-Norm	Desc	47	0.0348995	Unsorted	–	51	0.0419254
	5	2-Norm	Asc	33	0.1541805	Unsorted	–	67	0.0359197	Diagonals	Desc	64	0.0388210
22	1	Diagonals	Asc	47	0.1413312	2-Norm	Asc	43	0.0442212	1-Norm	Asc	32	0.0522907
	2	1-Norm	Desc	73	0.1015856	1-Norm	Asc	57	0.0418875	Diagonals	Desc	42	0.0478554
	3	1-Norm	Asc	63	0.1750429	Infinity	Desc	65	0.0422199	2-Norm	Desc	68	0.0506885
	4	Frobenius	Desc	35	0.1676817	Unsorted	–	47	0.04038048	2-Norm	Asc	32	0.0519724
	5	1-Norm	Desc	62	0.1233559	Frobenius	Desc	43	0.0414922	1-Norm	Desc	44	0.0513620
24	1	2-Norm	Desc	54	0.1429453	2-Norm	Asc	45	0.0431852	Infinity	Asc	36	0.0596867
	2	Infinity	Asc	41	0.1293378	Unsorted	–	59	0.0385926	2-Norm	Desc	57	0.0737224
	3	Infinity	Asc	40	0.1985698	1-Norm	Desc	69	0.0513072	Diagonals	Asc	57	0.0619988
	4	1-Norm	Desc	51	0.2304580	2-Norm	Desc	67	0.0448926	Infinity	Asc	36	0.0613723
	5	1-Norm	Asc	25	0.2360208	Unsorted	–	46	0.0456920	Diagonals	Desc	48	0.0615945
26	1	Unsorted	–	44	0.1410162	Frobenius	Desc	40	0.0498521	Infinity	Desc	43	0.0683939
	2	Unsorted	–	66	0.1382613	2-Norm	Desc	60	0.0530870	Frobenius	Desc	59	0.0679481
	3	1-Norm	Desc	60	0.1403570	Frobenius	Desc	46	0.0462112	Diagonals	Asc	67	0.0696464
	4	Diagonals	Desc	43	0.1330237	Diagonals	Desc	59	0.0493228	Unsorted	–	51	0.0611022
	5	Frobenius	Desc	26	0.1785924	Diagonals	Desc	50	0.0518575	Frobenius	Desc	37	0.0625734

TABLE 17. List of proposed heuristics for 3IAP.

	DM	GSP	Category 1	Category 2
Sorting criteria	Diagonals	–	Statistical	Matrix norms
	Diagonals factor		Mean	Frobenius
			Median	1-norm
			Maximum	Infinity
			Range	2-norm
			Std. Deviation	

TABLE 18. Computational results of hybridized DM on Sample 1.

Instances	B&B		DM		Hybrid DM	
Size n	Cost	Time	Cost	Time	Cost	Time
4	49.42	0.2530	94.28	0.0164	49.42	0.0490
6	36.64	0.3290	90.76	0.0128	36.64	0.1893
8	24.42	0.9430	99.00	0.0229	24.42	0.7347
10	20.79	2.5689	112.38	0.0266	20.79	2.2989
12	21.28	7.4436	95.31	0.0315	21.28	7.2635
14	18.05	17.5816	111.58	0.0361	18.05	17.2285
16	19.43	31.2333	105.40	0.0409	19.43	30.6277
18	17.91	126.8548	114.23	0.0347	17.91	121.5693
20	12.74	51.6672	97.32	0.0614	12.74	50.1720
22	13.28	445.9688	151.39	0.0608	13.28	429.8912
24	10.84	1599.6080	125.39	0.1231	10.84	1410.3500
26	11.36	2956.9040	100.29	0.0878	11.36	2894.0040

TABLE 19. Comparison of the Gurobi outcomes with Balas and Saltzman's results [8].

	B & S		Gurobi	
Size	Cost	CPU	Cost	CPU
4	42.20	0.03	42.20	0.266514
6	40.20	0.16	40.20	0.055449
8	23.80	0.84	23.80	0.142794
10	19.00	1.36	19.00	0.353109
12	15.60	2.19	15.60	0.953436
14	10.00	11.95	10.00	2.552104
16	10.00	39.90	10.00	5.050227
18	6.40	55.30	6.40	10.05813
20	4.80	169.29	4.80	19.00167
22	4.00	371.52	4.00	33.68928
24	1.80	514.52	1.80	55.62736
26	1.30	624	1.00	89.94152

The extensive computational tests indicate that the best solutions found are of good quality compared to DM results. Again, if the found solution is optimal or near-optimal, it will be accepted; otherwise, it will be used as a warm start for the B&B to produce a better solution.

TABLE 20. Gurobi results using Sample 2.

Size	Instance	Cost	Runtime
5	1	53	0.0654907
	2	25	0.0543890
	3	44	0.0583537
	4	22	0.0469968
	5	495	0.0564752
10	1	29	0.3719521
	2	37	0.3552456
	3	29	0.3597858
	4	33	0.3668726
	5	321	0.3674960
15	1	37	3.4365306
	2	23	3.5693522
	3	26	3.3586519
	4	24	3.2766328
	5	276	3.4546936
20	1	23	49.1807597
	2	24	49.0274122
	3	24	50.2407401
	4	23	49.8226774
	5	343	49.9715149
25	1	251	189.5268292
	2	251	186.6024799
	3	250	187.0178716
	4	250	189.8129685
	5	363	185.8806562
30	1	300	205.6796706
	2	300	212.9857600
	3	300	336.3311508
	4	300	562.0314245
	5	362	563.5667758
35	1	350	1,453.9072645
	2	350	1,443.5788038
	3	350	1,426.9249260
	4	350	1,441.0174892
	5	395	1,452.7928391

5.6. Exact solution approaches

5.6.1. Branch & Bound and Hybrid DM

Heuristics from both Category 1 and Category 2 consistently provide feasible solutions for 3IAP, with a possibility of optimality. When the discrepancy between the linear program lower bound and its dual upper bound is zero or negligible, the identified solution is deemed optimal or near-optimal, respectively. In cases where this gap is significant, the derived solution serves as a warm start for the B&B method. The integration of DM with B&B leads to optimal or nearly optimal outcomes, outperforming the standard B&B approach as Table 18 demonstrates.

Upon conducting numerical experiments on Sample 1 dataset, [25] observed that a hybridized DM reaches optimal costs akin to the B&B method in significantly reduced computation time, provided the problem size does

TABLE 21. Comparison of Gurobi and MIP performances.

	Gurobi		MIP		
Size	Cost	Time	Cost	Time	% Saved time
4	42.20	0.0266514	42.20	0.0329738	–
6	40.20	0.0554495	40.20	0.0498471	10%
8	23.80	0.1427943	23.80	0.0893315	37%
10	19.00	0.3531094	19.00	0.1383330	61%
12	15.60	0.9534360	15.60	0.2144092	78%
14	10.00	2.5521043	10.00	0.4166004	84%
16	10.00	5.0502273	10.00	0.7390756	85%
18	6.40	10.0581329	6.40	1.2691315	87%
20	4.80	19.0016731	4.80	2.3635161	88%
22	4.00	33.6892783	4.00	3.7201386	89%
24	1.80	55.6273630	1.80	5.7691557	90%
26	1.00	89.9415218	1.00	8.9291064	90%

not exceed 26. The author reported that any attempt to implement B&B on larger instances has consistently failed, highlighting a limitation in scalability for larger problems.

After developing and implementing heuristic methods for 3IAP that share the DM structure, it is essential to assess their performance. Their evaluation should include comparing the heuristic solutions with those generated by exact algorithms to ascertain their effectiveness.

5.6.2. Gurobi optimizer

For the optimal resolution of 3IAP, a computer program integrating the *Gurobi Optimization* solver [21], is developed. The Gurobi Optimizer, renowned for its cutting-edge capabilities in mathematical programming, employs the latest in algorithms and architectures. The program refers to the second 3IAP mathematical formulation LP2, implemented through the Python Notebook interface for Gurobi optimization.

The program performance is assessed using Sample 1 as the benchmark dataset. Table 19 presents the outcomes of the Gurobi optimizer compared to the results of [8]. Each table row outlines the average cost and the average execution time of five instances of the same size. The findings indicate that Gurobi achieves optimal solutions comparable to those of [8] but with reduced CPU time.

Let Sample 2 be a second random sample generated by uniform probability from 0 to 100, of size $n = 5, 10, 15, \ldots, 35$; five instances per size except the fifth instance from 100 to 999. Numerical experiments are conducted on Sample 2 and the results are presented in Table 20.

The performance of the proposed program is evaluated through computational tests on larger instances, utilizing a random sample of sizes up to 35. The program attained optimal solutions for these instances, demonstrating efficiency with a satisfactory runtime. Similar computational testing could be performed on instances of larger sizes but obviously for a longer execution time.

5.6.3. Python-MIP solver

After implementing and testing a first program incorporating the Gurobi optimizer, a second one was developed for solving optimally 3IAP, using a specific package Python-MIP, an advanced library for Mixed-Integer Linear Programming problems, facilitating their modelling and resolution.

Two computing researchers [56] designed MIP a fast open source which works with the Gurobi Optimization solver. Another computer program was proposed to solve 3IAP optimally by adapting the MIP solver through the Python programming language and using the first mathematical formulation LP1 of 3IAP.

In Table 21 presents a comparative analysis between the Gurobi solver and the Python-MIP program, utilizing the same dataset, namely Sample 1. Each row in the table summarizes the average results of five instances of the

TABLE 22. Comparing Gurobi and MIP results using Sample 2 data.

		Gurobi		MIP	
Size	Instance	Cost	Runtime	Cost	Runtime
5	1	53	0.0654907	53	0.0546420
	2	25	0.0543890	25	0.0594075
	3	44	0.0583537	44	0.0614631
	4	22	0.0469968	22	0.0651166
	5	495	0.0564752	495	0.0731392
10	1	29	0.3719521	29	0.1645260
	2	37	0.3552456	37	0.1124990
	3	29	0.3597858	29	0.1082377
	4	33	0.3668726	33	0.1196458
	5	321	0.3674960	321	0.1081557
15	1	37	3.4365306	37	0.5428748
	2	23	3.5693522	23	0.5998354
	3	26	3.3586519	26	0.5898159
	4	24	3.2766328	24	0.5886486
	5	276	3.4546936	276	0.5112021
20	1	23	49.1807597	23	2.1273868
	2	24	49.0274122	24	2.3294091
	3	24	50.2407401	24	2.2434087
	4	23	49.8226774	23	2.2193303
	5	343	49.9715149	343	2.4717333
25	1	251	189.5268292	251	7.3689439
	2	251	186.6024799	251	7.2044478
	3	250	187.0178716	250	7.2811375
	4	250	189.8129685	250	7.4009490
	5	363	185.8806562	363	7.1637032
30	1	300	205.6796706	300	19.9615598
	2	300	212.9857600	300	19.9107783
	3	300	336.3311508	300	20.1751885
	4	300	562.0314245	300	20.6488466
	5	362	563.5667758	362	19.9449713
35	1	350	1453.9072645	350	48.5529411
	2	350	1443.5788038	350	47.7848966
	3	350	1426.9249260	350	48.7666168
	4	350	1441.0174892	350	48.9653962
	5	395	1452.7928391	395	50.6136987

same size. The last column shows the percentage reduction in CPU time. The Python-MIP program performance surpasses that of the Gurobi solver, as evidenced by the savings in computational time.

The Python-MIP program implementation demonstrates excellent outcomes, producing optimal solutions identical to those generated by Gurobi but with significantly reduced runtime. Computational tests conducted on Sample 2 are detailed in Table 22, which compares the results from Gurobi with those of the Python-MIP program. Figure 4 elucidates the average computation times for both Gurobi and MIP across five instances of the same size, ranging from $n = 5$ to 35, indicating an enhanced efficiency of the MIP program.

More numerical tests were conducted on random data of Sample 3, which is Sample 2 extended to the size $n = 60$. Sample 3 is composed of five instances of size $n = 35, 40, 45, \ldots, 60$, generated by uniform probability between 10 and 999, and the test results are displayed in Table 23. Table 24 summarises the algorithms execution time for different random samples. The Python-MIP performance exceeds that of the proposed heuristics and

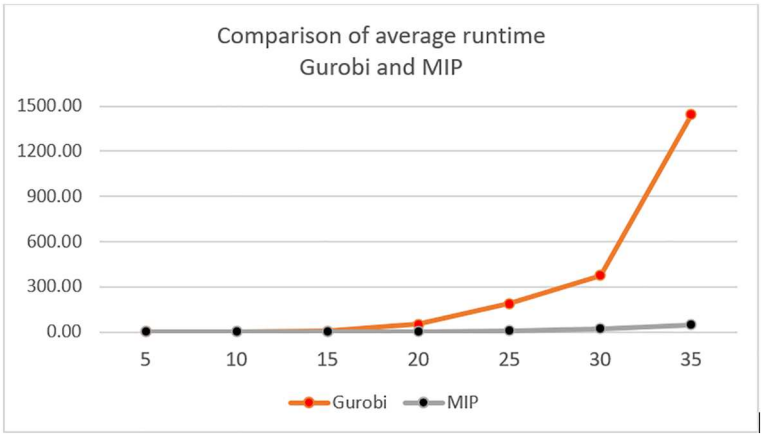

FIGURE 4. Average runtime of Gurobi and MIP.

TABLE 23. MIP outcomes using random data of Sample 3.

MIP results using Sample 3			
Size	Instance	Cost	Runtime
35	1	350	48.5529411
	2	350	47.7848966
	3	350	48.7666168
	4	350	48.9653962
	5	395	50.6136987
40	1	400	104.3042250
	2	400	103.7430441
	3	400	104.4073768
	4	400	106.0345357
	5	463	111.4614673
45	1	450	211.8499980
	2	450	210.6143940
	3	450	210.8913579
	4	450	218.1257191
	5	489	288.3644428
50	1	500	417.2589750
	2	500	445.6542146
	3	500	1,141.8830636
	4	500	1,121.1757991
	5	532	1,474.7669525
55	1	550	930.1332147
	2	550	754.0068917
	3	550	715.4535873
	4	550	708.2575543
	5	583	1,013.1942210
60	1	600	1,223.0954397
	2	600	1,225.4023554
	3	600	1,211.7568259
	4	600	1,221.9468393
	5	619	3,466.8056731

Table 24. Runtime in minutes.

		Gurobi	MIP	
	Size	Average runtime (in min)		Time saved (%)
Sample 1	26	1.499	0.149	90
Sample 2	35	24.061	0.816	97
Sample 3	60	–	27.830	–

Gurobi program, as its execution time is significantly reduced from 10% to 90%. For example, when running 3IAP instances of size $n = 26$ and $n = 35$, Gurobi takes about 1.5 min and 24 min, respectively while MIP requires only 0.15 min and less than 1 min, equivalent to 90% and 97% reduction in running time, respectively. Hence, when executing 3IAP instances of size $n = 60$, the Python-MIP program running time is less than 28 min. Further tests are planned for larger instances.

6. Conclusion

The paper investigates 3IAP, an extended version of the standard two-dimensional assignment problem LAP. The GSP and DM are proposed for solving 3IAP, they have been expanded by considering tie-cases leading to the development of three variants. Furthermore, by exploring specific characteristics of the cost matrices, two novel heuristic classes are suggested for solving 3IAP. The first class relies on the statistical measures of cost matrices, while the second refers to matrix norms.

After extensive computational testing, the results indicate that the proposed heuristics can yield feasible solutions in polynomial time. If the resulting solutions are of high quality, they will be retained; otherwise, they can be used as initial values for the B&B method to produce better solutions. Finally, two computer programs, Gurobi and Python-MIP, are implemented for comparison purposes, achieving optimal solutions for large instances in competitive computational time. Both programs produced optimal solutions, compared with benchmark data and those randomly generated, but the MIP performance far outstrips that of Gurobi.

The literature indicates that the largest 3IAP instance solved to optimality is of size $n = 26$. By implementing Gurobi and MIP programs using their Python interface, we have optimally solved 3IAP instances of sizes $n = 35$, $n = 60$ and potentially higher sizes. Future research is planned to explore the tie-cases further to find solutions for 3IAP of larger sizes and resolve at least two concrete 3IAP applications.

Acknowledgments

The author would like to thank the anonymous reviewers for their helpful comments, feedback and their valuable suggestions and he thanks Dr. Nouredin Sadawi, a computing expert, for accepting to review the paper code.

Funding

This research did not receive any funding agencies in the public, commercial, or not-for-profit sectors.

Conflicts of interest

The author declares no conflicts of interest.

Data Availability Statement

This paper's data and essential components of its code are available online in Github repository, https://github.com/mmehbali/Three-IAP [40]. There are two folders, one contains three samples of data randomly generated and the other includes the main parts of the code. Further details can be provided to potential researchers upon request.

References

[1] L.G. Afraimovich and M.D. Emelin, Combining solutions of the axial assignment problem. *Autom. Remote Control* **82** (2021) 1418–1425.

[2] L.G. Afraimovich and M.D. Emelin, Complexity of solutions combination for the three-index axial assignment problem. *Mathematics* **10** (2022) 1062.

[3] G. Appa, D. Magos and I. Mourtos, On multi-index assignment polytopes. *Linear Algebra Appl.* **416** (2006) 224–241.

[4] A. Arora, *Multiple alignment of protein interaction networks by three-index assignment algorithm*, MSc. thesis in Computer Science, University of Windsor, Ontario, Canada (2013).

[5] E. Balas and L. Qi, Linear-time separation algorithms for the three-index assignment polytope. *Discrete Appl. Math.* **43** (1993) 1–12.

[6] E. Balas and P.R. Landweer, Traffic assignment in communication satellites. *Oper. Res. Lett.* **2** (1983) 141–147.

[7] E. Balas and M.J. Saltzman, Facets of the three-index assignment polytope. *Discrete Appl. Math.* **23** (1989) 201–229.

[8] E. Balas and M.J. Saltzman, An algorithm for the three-index assignment problem. *Oper. Res.* **39** (1991) 150–161.

[9] R.E. Burkard, R. Rudolf and G.J. Woeginger, Three-dimensional axial assignment problems with decomposable cost coefficients. *Discrete Appl. Math.* **65** (1996) 123–139.

[10] Y. Crama and F.C.R. Spieksma, Approximation algorithms for three-dimensional assignment problems with triangle inequalities. *Eur. J. Oper. Res.* **60** (1992) 273–279.

[11] T. Dokka, A.N. Letchford and M.H. Mansoor, *Using surrogate relaxation as a matheuristic*. Internal report, Department of Management Science, Lancaster University, Lancaster, UK (2021).

[12] T. Dokka and F.C.R. Spieksma, Facets of the axial three-index assignment polytope. *Discrete Appl. Math.* **201** (2016) 86–104.

[13] T. Dokka, Y. Crama and F.C.R. Spieksma, Multi-dimensional vector assignment problems. *Discrete Optim.* **14** (2014) 111–125.

[14] R. Euler, Odd cycles and a class of facets of the axial 3-index assignment polytope. *Appl. Math.* **19** (1987) 375–386.

[15] S. Faudzi, S. Abdul-Rahman and R. Abd Rahman, An assignment problem and its application in education domain: a review and potential path, Hindawi (2018). DOI: 10.1155/2018/8958393.

[16] A.M. Frieze and J. Yadegar, An algorithm for solving 3-dimensional assignment problems with application to scheduling a teaching practice. *J. Oper. Res. Soc.* **32** (1981) 989–995.

[17] B. Gabrovšek, T. Novak, J. Povh, D.R. Poklukar and J. Žerovnik, Multiple Hungarian method for k-assignment problem. *Mathematics* **8** (2020) 2050.

[18] M.R. Garey and D.S. Johnson, Computers and Intractability. Freeman, San Francisco (1979).

[19] H. Gauvrit, J. Le Cadre and C. Jauffret, formulation of multitarget tracking as an incomplete data problem. *IEEE Trans. Aerosp. Electron. Syst.* **33** (1997) 1242–1257.

[20] D. Grundel, R. Murphey, P.M. and Pardalos, Theory and Algorithms for Cooperative Systems. World Scientific, River Edge (2004).

[21] Gurobi (2024). https://www.gurobi.com/.

[22] J. He, S. Zhang, Z. Ren, X. Lai and Y. Piao, Approximate muscle guided beam search for three-index assignment problem. In: *Proceedings of the 5th International Conference on Swarm Intelligence* (2014). https://doi.org/10.48550/arXiv.1703.01893.

[23] G. Huang and A. Lim, A hybrid genetic algorithm for three-index assignment problem. In: *Proceedings of the 2003 IEEE Congress on Evolutionary Computation, CEC 2003* (2003) 2762–2768.

[24] M. Johnsson, G. Magyar and O. Nevalainen, On the Euclidean 3-matching problem. *Nordic J. Comput.* **5** (1998) 143–171.

[25] D. Kadhem, *Heuristic solution approaches to the solid assignment problem*, Ph.D. thesis, University of Essex, UK (2017).

[26] A.R. Kammerdiner and C.F. Vaughan, Very large-scale neighborhood search for the multidimensional assignment problem, in Optimization Methods and Applications. Springer, Cham (2017) 251–262.

[27] A.R. Kammerdiner, A. Semenov and E. Pasiliao, Landscape properties of the very large-scale and the variable neighborhood search metaheuristics for the multidimensional assignment problem (2021). Preprint arXiv:2112.11849.

[28] A.R. Kammerdiner, A. Semenov and E.L. Pasiliao, Multidimensional assignment problem for multipartite entity resolution. *J. Glob. Optim.* (2022) 1–33.

[29] D. Karapetyan and G. Gutin, Local search heuristics for the multidimensional assignment problem. *J. Heuristics* **17** (2011) 201–249.

[30] D. Karapetyan and G. Gutin, A new approach to population sizing for memetic algorithms: a case study for the multidimensional assignment problem. *Evol. Comput.* **19** (2011) 345–371.

[31] R.M. Karp, Reducibility among combinatorial problems. In: *Complexity of Computer Computations*. The IBM Research Symposia Series, Springer, Boston (1972) 85–103.

[32] V.M. Kravtsov, Description of the types of maximum non-integer vertices of the polyhedron in the three-index axial assignment problem. *Comput. Math. Math. Phys.* **46** (2006) 1821–1825.

[33] V.M. Kravtsov, Combinatorial properties of non-integer vertices of a polytope in a three-index axial assignment problem. *Cybern. Syst. Anal.* **43** (2007) 25–33.

[34] H.W. Kuhn, The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **2** (1955) 83–97.

[35] Y. Kuroki and T. Matsui, An approximation algorithm for multidimensional assignment problems minimizing the sum of squared errors. *Discrete Appl. Math.* **157** (2009) 2124–2135.

[36] J. Li, R. Tharmarasa, D. Brown, T. Kirubarajan and K.R. Pattipati, A novel convex dual approach to three-dimensional assignment problem: theoretical analysis. *Comput. Optim. Appl.* **74** (2019) 481–516.

[37] G. Magyar, M. Johnsson and O. Nevalainen, An adaptive hybrid genetic algorithm for the three-matching problem. *IEEE Trans. Evol. Comput.* **4** (2000) 135–146.

[38] A.R. Mahjoub, Two-edge connected spanning subgraphs polyhedra. *Math. Program.* **64** (1994) 199–208.

[39] A.R. Mahjoub, Polyhedral approaches, in Concepts of Combinatorial Optimization. John Wiley & Sons, Inc. (2014) 261–324.

[40] S.N. Medvedev and O.A. Medvedeva, An adaptive algorithm for solving the axial three-index assignment problem. *Autom. Remote Control* **80** (2019) 718–732.

[41] M. Mehbali (2025). https://github.com/mmehbali/Three-IAP.

[42] J. Munkres, Algorithms for the assignment and transportation problems. *J. Soc. Ind. Appl. Math.* **5** (1957) 32–38.

[43] S. Pal, I. Khan and M.K. Maiti, A heuristic approach to solve multidimensional assignment problem, in Proceedings of the 2018 4th International Conference on Recent Advances in Information Technology (RAIT). IEEE (2018) 1–7.

[44] E.L. Pasiliao, P.M. Pardalos and L.S. Pitsoulis, Branch and bound algorithms for the multidimensional assignment problem. *Optim. Methods Softw.* **20** (2005) 127–143.

[45] S.L.P. Perez, *Personnel assignment problems through the multidimensional assignment problem*. Ph.D. thesis on Optimization Universidad Autónoma Metropolitana, Mexico City, Mexico (2017).

[46] W.P. Pierskalla, The tri-substitution method for the multidimensional assignment problem. *Can. Oper. Res. Soc.* **5** (1967) 71–81.

[47] W.P. Pierskalla, Letters to the editor - the multidimensional assignment problem. *Oper. Res.* **16** (1968) 422–454.

[48] A.B. Poore, Multidimensional assignment problems arising in multitarget and multisensor tracking. In: *Nonlinear Assignment Problems*. Springer, Boston (2000) 13–38.

[49] A.B. Poore and A.J. Robertson, A new Lagrangian relaxation based algorithm for a class of multidimensional assignment problem. *Comput. Optim. Appl.* **8** (1997) 129–150.

[50] A.B. Poore and S. Gadaleta, Some assignment problems arising from multiple target tracking. *Math. Computer Model.* **43** (2006) 1074–1091.

[51] A.B. Poore, S. Lu and B.J. Suchomel, Data association using multiple-frame assignments. In: *Handbook of Multisensor Data Fusion*. CRC Press, Boca Raton (2017) 319–338.

[52] L. Qi, L. and D. Sun, Polyhedral methods for solving three index assignment problems. In: *Nonlinear Assignment Problems*. Springer, Boston (2000) 91–107.

[53] L.E.U. Rivero, M.R.B. Escarcega and J. Velasco, An integer linear programming model for a case study in classroom assignment problem. *Comput. Sist.* **24** (2020) 97–104.

[54] F.C.R. Spieksma, Multi-index assignment problems: complexity, approximation, applications. In: *Nonlinear Assignment Problems*. Springer, Boston (2000) 1–12.

[55] F.C.R. Spieksma and G.J. Woeginger, Geometric three-dimensional assignment problems. *Eur. J. Oper. Res.* **91** (1996) 611–618.

[56] T.A.M. Toffolo and H.G. Santos, Introduction - Python-MIP documentation (2019). https://docs.python-mip.com/en/latest/intro.html.

[57] S. Vadrevu and R. Nagi, A dual-ascent algorithm for the multi-dimensional assignment problem: application to multi-Target tracking. In: *Proceedings of the 2020 IEEE 23rd International Conference on Information Fusion (FUSION)*. IEEE (2020) 1–8.

[58] C.E. Valencia, F.J.Z. Martinez, S.L.P. and Perez, A simple but effective memetic algorithm for the multidimensional assignment problem. In: *Proceedings of the 2017 14th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*. IEEE (2017).

[59] C.E. Valencia, C.A. Alfaro, F.J.Z. Martinez, M.C.V Magana and S.L.P. Perez, Outperforming several heuristics for the multidimensional assignment problem. In: *Proceedings of the 2018 15th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*. IEEE (2018).

[60] C.J. Veenman, M.J.T. Reinders and E. Backer, Establishing motion correspondence using extended temporal scope. *Artif. Intell.* **145** (2003) 227–243.

[61] E.R. Wulan, D. Jamaluddin and W.N. Ramadhan, Determining optimal solutions in learning outcome using one to one fixed method. In: *Proceedings of the International Conference on Science and Engineering (ICSE-UIN-SUKA 2021)*. Atlantis Press (2021) 7–11.

[62] L. Zhang, D. Sidoti, S. Vallabhaneni, K.R. Pattipati and D.A. Castnon, Approaches to obtain a large number of ranked solutions to 3-dimensional assignment problems. *J. Adv. Inf. Fusion* **13** (2017) 50–65.

[63] Y. Zhang, Y. Li, S. Li, J. Zeng, Y. Wang and S. Yan, Multi-target tracking in multi-static networks with autonomous underwater vehicles using a robust multi-sensor labeled Bernoulli filter. *J. Mar. Sci. Eng.* **11** (2023) 875.