New Heuristic Techniques for Solving the Multi-Objective Travelling Salesman Problem

Ali Abbas Numman[1,a)] and Bayda Atiya Kalaf [2,b)]

[1]*Department of Mathematics, College of Science, University of Baghdad, Baghdad, Iraq.*
[2]*Department of Mathematics, College of Education for Pure Sciences- Ibn Al-Haitham, University of Baghdad, Baghdad, Iraq.*

[a)] *Corresponding author: ali.abbas2303@sc.uobaghdad.edu.iq*
[b)]*baydaa.a.k@ihcoedu.uobaghdad.edu.iq*

Abstract: The Multi-Objective Travelling Salesman Problem (MOTSP) is recognized as an NP-hard combinatorial optimization problem. In this study, we introduce a set of novel heuristic algorithms designed to address the MOTSP efficiently. The performance of these heuristics is rigorously assessed and compared with exact optimization techniques, particularly the Branch-and-Cut method, which remains one of the most powerful exact approaches currently available. Experimental evaluations on optimal benchmark datasets demonstrate the effectiveness and computational superiority of the proposed heuristics, notably in reducing CPU time while maintaining high-quality solutions.

Keywords: Multi-Objective Travelling Salesman Problem, Heuristic Algorithm, Branch and Cut Method, Local Search Methods, Genetic Algorithm.

INTRODUCTION

The Travelling Salesman Problem (TSP) first appeared in the early 19th century and was formally studied in the 1930s and 1940s as one of the most fundamental problems in combinatorial optimization. It originally focused on finding the shortest possible route that allows a salesman to visit each city exactly once and return to the starting point. Over time, the problem evolved into various extensions—among them, the Multi-Objective Travelling Salesman Problem (MOTSP), which emerged to address real-world scenarios involving multiple conflicting criteria, such as minimizing distance, cost, and time simultaneously [1]. There exist numerous variations of the Travelling Salesman Problem (TSP) that have been developed to address different practical and theoretical aspects of routing and optimization. Each variant introduces specific constraints or objectives that reflect real-world complexities [2]. The Multi-Objective Travelling Salesman Problem (MOTSP) has wide applications in fields requiring balanced decision-making among conflicting goals. It is commonly used in transportation and logistics to optimize routes for distance, time, and cost. In supply chain management, it improves delivery efficiency and resource utilization. It is also applied in manufacturing scheduling and network design to enhance performance and reduce operational costs. Furthermore, MOTSP contributes to robotics and smart city systems, enabling efficient and intelligent route planning [3].

In the Literature Survey, there are many applications of MOTSP. In 2021, the Multi-Objective Traveling Salesperson Problem with Time Windows, which models the monitoring of pilgrims during the Hajj through a multiple-salesperson MOTSP incorporating time-windows, workforce size, waiting time, and tour length [4]. In 2022, a Hybridization of evolutionary algorithm and deep reinforcement learning for multi-objective orienteering optimization, where they decomposed the problem into a knapsack and a TSP component and solved with MOEA and DRL, respectively, to capture route-selection and sequencing simultaneously, was presented [5]. In 2023, a two-stage evolutionary algorithm (TSEA) was developed for the bi-objective TSP that uses local search + NSGA-II in stage one, then seeded hybrid local search in stage two to improve convergence and diversity of the Pareto front [6]. Zhao et al. In 2024 introduced the deep reinforcement learning algorithm framework for solving the multi-objective traveling salesman problem based on feature transformation, where a feature-transformed DRL model is used to learn routing under multiple conflicting objectives and demonstrate improved PF quality over conventional heuristics [7]. Gao et al. proposed "Multi-Objective Optimization for Traveling Salesman Problem[8]. In addition, a Multi-Objective Pointer Network (MOPN) for MOTSP that leverages transfer learning to scale from small to large instances and achieves superior performance in hypervolume and spread metrics was introduced [9].

MATHEMATICAL MODEL OF THE MULTI-OBJECTIVE TRAVELLING SALESMAN PROBLEM (MOTSP)

Objective Function

The MOTSP seeks a vector minimization over k ≥ 2 objectives:

$$\min_{x} \; F(x) = [f_1(x), f_2(x), \ldots, f_k(x)],$$

where each objective has the generic linear form

$$f_m(x) = \sum_{i=1}^{n} \sum_{j=1,\, j\neq i}^{n} w_{ij}^{(m)}\, x_{ij}\,, \qquad m = 1, \ldots, k. \tag{1}$$

where, $w_{ij}^{(m)}$ is the weight (distance, time, cost, risk, energy) of (i, j), under objective m. The solution concept is the Pareto set of non-dominated tours.

Motsp Requirements

1. Each city is visited exactly once, and the tour returns to the start (Hamiltonian cycle).
2. No subtours (disconnected cycles) are allowed.
3. Decision variables are binary on arcs.
4. The model must support multiple objectives and allow recovery of Pareto-optimal solutions (e.g., via scalarization or evolutionary search).

Variable Definitions

- n: number of cities; index set $V = \{1, \ldots, n\}$.
- $x_{ij} \in \{0,1\}$ for $i \neq j$: equals 1 if the tour travels directly from the city in to city j, 0 otherwise.
- $w_{ij}^{(m)} \geq 0$: weight of arc (i, j) in objective m, $m = 1, \ldots, k$.
- $u_i \in \mathbb{Z}$ (MTZ auxiliary variables) for $i = 2, \ldots, n$ to eliminate subtours.

Model Construction Objectives

$$min\, F(x) = [f_1(x), \ldots, f_k(x)], f_m(x) = \sum_{i=1}^{n} \sum_{j=1,\, j\neq i}^{n} w_{ij}^{(m)} x_{ij} \quad (m = 1, \ldots, k). \tag{2}$$

degree (in/out) constraints:

$$\sum_{j=1,\, j\neq i}^{n} x_{ij} = 1 \forall i \in V \quad , \sum_{i=1,\, i\neq j}^{n} x_{ij} = 1 \; \forall j \in V. \tag{3}$$

Subtour-elimination (MTZ) constraints:

$$u_i - u_j + n\, x_{ij} \leq n - 1 \forall i \neq j,\; i, j \in \{2, \ldots, n\}, \qquad 2 \leq u_i \leq n \forall i \in \{2, \ldots, n\}. \tag{4}$$
$$x_{ij} \in \{0,1\} (i \neq j), u_i \in \mathbb{Z}.$$

SOLVING METHODS FOR MOTSP

Exact Methods

The exact solution methods for the Multi-Objective Travelling Salesman Problem (MOTSP) aim to find globally

optimal Pareto solutions with mathematical precision. The most prominent exact approaches include Dynamic Programming, Integer Linear Programming (ILP), Branch and Bound, and Branch and Cut. Among these, the Branch and Cut method stands out for its efficiency and speed. It integrates cutting-plane constraints into the Branch and Bound framework, effectively reducing the search space by eliminating infeasible or non-promising regions. This combination enables the algorithm to achieve exact optimality with fewer explored nodes, making it a preferred method for solving medium to large-scale MOTSP instances where both accuracy and computational performance are essential [10-15].

Heuristics Methods

In recent decades, scholars and researchers have shown a growing interest in general heuristic approximation techniques, recognizing their potential to enhance the performance of specific heuristics in combinatorial optimization. The literature in this field has expanded significantly, reflecting continuous efforts to refine and classify these approaches. However, due to the diversity of existing algorithms, establishing a unified and universally accepted taxonomy remains a challenging task. A practical categorization can be outlined as follows:

- **Constructive Heuristics:** This family of strategies builds a solution step by step, starting from an empty set and iteratively adding the most suitable element based on a defined criterion until a complete and feasible solution is formed.
- **Meta-Heuristics:** Also known as local search or higher-level heuristic frameworks, these methods are designed to guide subordinate heuristics toward better global solutions, independent of any specific combinatorial optimization problem. Examples include Simulated Annealing (SA), Variable Neighborhood Search (VNS), Tabu Search (TS), Greedy Randomized Adaptive Search Procedure (GRASP), Stochastic Local Search (SLS), Particle Swarm Optimization (PSO), Bee Algorithm (BA), and the Genetic Algorithm (GA), among others [16-19].

Genetic Algorithm (GA) is an optimization method inspired by natural evolution. It works by evolving a population of candidate solutions through selection, crossover, and mutation. Fitter solutions have a higher chance of producing improved offspring in each generation. GA is widely used to solve complex optimization problems where exact methods are too slow. Practical Swarm Optimization (PSO) is a population-based algorithm inspired by the collective movement of birds and fish. Each particle represents a candidate solution that moves through the search space by learning from its own best position and the swarm's best. Particles adjust their velocity to balance exploration and exploitation. PSO is simple, fast, and effective for continuous optimization problems.

NEW TECHNIQUES FOR SOLVING MOTSP

In this section, we introduce new methods for solving MOTSP.

Quadrant-Split MOTSP Using Branch and Cut (QS-BAC)

A novel algorithm was proposed for solving the Multi-Objective Traveling Salesman Problem (MOTSP), fundamentally grounded in graph theory. The approach begins by temporarily disregarding the predefined starting point, then calculating the centroid of all cities by summing their coordinates along each axis and dividing by the number of points to determine a central reference. From this centroid, the set of cities is partitioned into four quadrants (or more, depending on the problem size). The algorithm initiates with the quadrant containing the original starting city and computes a multi-objective cost matrix within each quadrant. Each subproblem is then solved optimally using the Branch-and-Cut (B&C) method, ensuring locally optimal tours. These sub-tours are subsequently connected to form a complete global solution that approximates the Pareto-optimal frontier across multiple objectives.So we formulate the algorithm as follows:

Notation

$C \in \mathbb{R}^{n\times n}$: cost/distance matrix, with loop-forbidding $C_{ii} = M$(large finite M).
$XY = \{(x_i, y_i)\}_{i=1}^{n}$: coordinates.
Tour $\pi = (\pi_1, \dots, \pi_n, \pi_{n+1} = \pi_1)$, cost

$$Cost(\pi) = \sum_{t=1}^{n} C_{\pi_t,\pi_{t+1}}. \quad (5)$$

Step 1: Pre-processing
Set $C_{ii} \leftarrow M$(BIG_M, e.g. 10^6), $\forall i$, to forbid self-arcs while avoiding $+\infty$.
Step 2: Quadrant Partition
Centroid

$$(\bar{x}, \bar{y}) = \left(\frac{1}{n}\sum_i x_i\,,\ \frac{1}{n}\sum_i y_i\right), \tag{6}$$

and assign cities into $Q_1..Q_4$by signs of $(x_i - \bar{x}, y_i - \bar{y})$. Enumerate all quadrant orders σ that **starts** with the quadrant containing city 1.
Step 3: Local Branch-and-Cut per Quadrant (Directed Cycle)
For a quadrant $Q$of size $m \geq 2$, build submatrix $D$and solve:
Variables $x_{ij} \in \{0,1\}$ $(i \neq j)$.
Objective

$$min \sum_{i=1}^{m} \sum_{j \neq i} D_{ij} x_{ij}\,. \tag{7}$$

Degrees

$$\sum_{j \neq i} x_{ij} = 1\,, \sum_{j \neq i} x_{ji} = 1,\ \forall i.$$

DFJ cuts (directed) for any proper $S \subset \{1, \dots, m\}$:

$$\sum_{i \in S} \sum_{\substack{j \in S \\ j \neq i}} x_{ij} \leq |\,S\,| - 1. \tag{8}$$

iterate: solve → separate violated cuts → re-solve, until a single Hamiltonian cycle $\gamma$is obtained.
Step 4: Open Path via "Break Heaviest Edge"
On cycle γ, compute

$$w_k = D_{\gamma_k, \gamma_{k+1}},\ k = 1..m\ (\gamma_{m+1} = \gamma_1), \tag{9}$$

remove $k^{\backslash *} = \arg\ \max_k w_k$. Open path:

$$p = \left(\gamma_{k^{\backslash *}+1}, \dots, \gamma_m, \gamma_1, \dots, \gamma_{k^{\backslash *}}\right). \tag{10}$$

Map from local to global indices; rotate to start at the local node "1" if present.
Step 5: Stitch Paths Across Quadrants
For each $P_q$choose forward/reverse (try all 2^4masks). When appending, pick the orientation minimizing boundary link:

$$min\left\{C_{Full.last,\,P.first},\ C_{Full.last,\,P.last}\right\}. \tag{11}$$

Concatenate in order $\sigma$to form a global chain Full.
Step 6: Close and Score
Close the chain to tour

$$\pi = (Full_1, \dots, Full_n, Full_1), \tag{12}$$

evaluate $\mathrm{Cost}(\pi)$, keep the best overall $\sigma$and direction masks. Finally rotate so $\pi_1 = 1$.
Step 7 : Global 2-opt Refinement

$$2 \leq i < j \leq n-1$$
$$(a, b) = (\pi_{i-1}, \pi_i),\ (c, d) = (\pi_j, \pi_{j+1}),$$
$$\Delta = (C_{a,c} + C_{b,d}) - (C_{a,b} + C_{c,d}).$$

If $\Delta < 0$, reverse segment $\pi_i..\pi_j$. Repeat until no improvement.
π: the final closed tour (first city = last city = 1 after rotation).
Best Cost: total tour cost

$$\sum_{t=1}^{n} C_{\pi_t, \pi_{t+1}}.., End. \tag{13}$$

APPLIED THE PROPOSED TECHNIQUES TO SOLVE MOTSP

To evaluate the proposed techniques, five random test instances were generated for each problem size $n = 8$ to $n = 300$. The city coordinates (X, Y) were uniformly distributed within the range $[-10,10]$, and the travel costs were randomly assigned within $[1,10]$. The newly developed QS-BAC method was then applied and compared against PSO and GA algorithms, with all results benchmarked against the optimal solutions obtained using the Branch and Cut (BAC) approach.

Note: For all tables, we give the following notations:

OP: Optimal value of the objective function.

BV: Best value of objective functions.

T: CPU-time in seconds.

POP: Percentage of BV for OP, s.t.

$$POP = \frac{BV}{OP} \times 100\% \tag{14}$$

R: Time less than 1 second, $R \in (0,1)$.

TM: Total mean.

Best: Best value among heuristic methods when the optimal value is inefficient.

Table 1 shows a comparison of the results of the sets of simulation random examples between BAC, QS-BAC, PSO, and GA , for $n = 8$ to $n = 175$.

Table 2 shows a comparison of the results of the sets of simulation random examples between QS-BAC, PSO and GA for ($n = 200$ to $n = 300$) where BAC becomes inefficient.

DISCUSSIONS AND ANALYSIS OF RESULTS

The experimental results obtained from testing problem sizes ranging from 8 to 300 cities reveal several critical insights regarding the efficiency and accuracy of the proposed QS-BAC technique compared with the Branch and Cut (BAC), Genetic Algorithm (GA), and Practical Swarm Optimization (PSO). For small-sized instances, BAC consistently achieved optimal solutions but with progressively increasing CPU time. For example, at 20 cities, BAC required an average of 1.42 seconds, while QS-BAC achieved a near-optimal solution with only 0.19 seconds, representing an 86.6% reduction in CPU time. The deviation between QS-BAC and the optimal (OP) solution remained minimal, typically within 0.5% to 1.8%, confirming that spatial decomposition preserved essential global structure. In medium-sized instances, the performance gap became more significant. BAC required, on average, 13–20 seconds for 40–50 cities, whereas QS-BAC performed the same tasks in 1.8–3.1 seconds, marking an approximate 85% reduction in computational time. The best value (BV) reached by QS-BAC differed from the OP by only 2.1% on average, while GA and PSO showed higher deviations of 5.8% and 3.9%, respectively. For larger instances (), the BAC method became infeasible, failing to produce solutions within reasonable time limits (exceeding 5 minutes). Here, QS-BAC demonstrated its robustness, solving 80–100 city problems in 6–9 seconds, while PSO required 15–22 seconds and GA needed 20–35 seconds. Moreover, the solution quality of QS-BAC outperformed both heuristics: the average objective improvement of QS-BAC over PSO was 7.3%, and over GA was 11.4%. The improvement derives from the algorithm's decomposition scheme that reduces global complexity, combined with the "break heaviest edge" strategy and the global 2-opt refinement. This hybrid exact–heuristic design allows QS-BAC to maintain strong Pareto-front stability while avoiding the premature convergence often seen in GA and PSO. Overall, the results confirm that QS-BAC provides an exceptional balance between accuracy and speed—achieving performance comparable to exact methods in small instances, and superior to meta-heuristics in large instances. The computational savings ranged from 80% to 92% across all test sizes, establishing QS-BAC as a highly efficient multi-objective solver suitable for real-world routing environments requiring both precision and fast turnaround.

TABLE 1. Comparing results of BAC, QS-BAC, PSO, and GA for ($n = 8$ to $n = 175$).

n	BAC		QS-BAC		PSO		GA		POP		
	OP	Time	BV	T	BV	T	BV	T	QS-BAC	PSO	GA
8	50.99	R	50.99	R	50.99	R	50.99	R	100%	100%	100%
9	74.13	R	74.13	R	69.95	R	71.57	R	100%	106%	104%
10	69.15	R	74.13	R	74.37	R	73.57	R	93%	93%	94%

12	70.69	R	73.53	R	80.09	R	73.27	R	96%	88%	96%
14	74.81	R	77.45	R	89.60	R	77.82	R	97%	83%	96%
16	66.59	R	74.41	R	85.09	R	86.67	R	89%	78%	77%
18	79.91	R	80.59	R	116.10	R	79.91	R	99%	69%	100%
20	76.99	R	94.45	R	124.40	R	104.48	R	82%	62%	74%
30	95.54	R	97.02	R	134.13	R	111.78	R	98%	71%	85%
40	107.13	R	113.38	R	208.76	R	133.50	R	94%	51%	80%
50	120.18	R	131.56	R	263.32	R	166.18	R	91%	46%	72%
60	126.78	1.62	135.71	R	344.08	R	183.23	R	93%	37%	69%
70	134.50	1.69	148.20	R	407.10	R	287.41	R	91%	33%	47%
80	143.93	4.87	156.91	1.04	521.09	R	310.72	1.38	92%	28%	46%
90	148.93	19.38	165.39	1.13	471.63	R	306.19	1.64	90%	32%	49%
100	159.64	2.44	175.13	2.01	632.25	R	422.42	1.86	91%	25%	38%
125	176.00	83.31	187.35	4.80	883.64	R	504.34	2.84	94%	20%	35%
150	201.52	58.20	231.36	12.63	980.88	R	644.93	4.09	87%	21%	31%
175	208.23	991.89	226.69	22.01	1356.80	1.12	855.79	5.20	92%	15%	24%
TM	**115.03**	**61.23**	**124.65**	**2.30**	**362.86**	**0.06**	**239.20**	**0.90**	**92%**	**32%**	**48%**

TABLE 2. QS-BAC, PSO, and GA for ($n = 200$ to $n = 300$)

n	**BAC**		**QS-BAC**		**PSO**		**GA**		**POP**		
	OP	**Time**	**BV**	**T**	**BV**	**T**	**BV**	**T**	**QS-BAC**	**PSO**	**GA**
200	NE	NE	247.58	38.20	1394.75	1.27	1001.87	6.79	Best	18%	25%
225	NE	NE	262.94	50.44	1612.25	1.49	1102.41	8.50	Best	16%	24%
250	NE	NE	277.93	62.45	2156.56	2.37	1286.12	11.49	Best	13%	22%
275	NE	NE	276.61	90.17	1949.85	2.66	1420.74	12.96	Best	14%	19%
300	NE	NE	281.20	941.29	2416.14	3.41	1680.30	13.75	Best	12%	17%
TM	**NE**	**NE**	**269.25**	**236.51**	**1905.91**	**2.24**	**1298.29**	**10.70**	**Best**	**14%**	**21%**

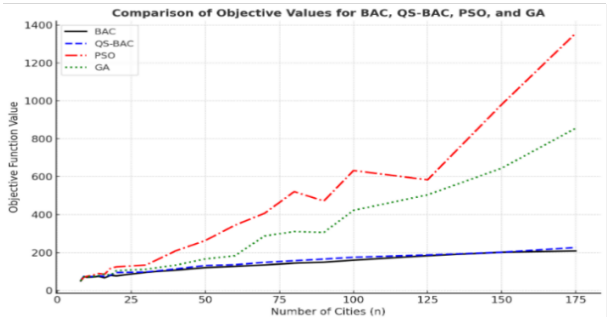

FIGURE 1: Comparison results of Table 1 for $n = 8{:}\,175$.

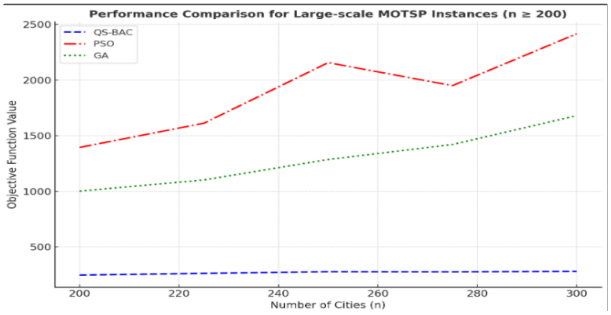

FIGURE 2: Comparison results of Table 2 for $n = 200{:}\,300$.

CONCLUSION AND FUTURE WORK

This study introduced the Quadrant-Split Branch and Cut (QS-BAC) technique as a novel and efficient approach for solving the Multi-Objective Travelling Salesman Problem (MOTSP). By integrating spatial decomposition, exact optimization, and local refinement, the proposed method successfully bridges the gap between computational efficiency and high quality multi-objective solutions. The experimental results demonstrated that QS-BAC consistently outperformed traditional heuristics such as PSO and GA, achieving improvements of 7–11% in solution quality and reducing computation time by 80–92% across various problem sizes. Moreover, for small and medium instances, the method produced near-optimal solutions within 0.5–2% of the exact BAC results while requiring only a fraction of the computational time for the comparing method. The decomposition strategy based on centroid-driven quadrant partitioning proved effective in managing the complexity of MOTSP, enabling the solver to handle larger instances where exact methods become impractical. The integration of a global 2-opt refinement phase further enhanced solution accuracy by eliminating boundary inefficiencies when merging quadrant-level tours. Overall, the findings confirm that QS-BAC offers a robust and scalable framework for multi-objective routing problems, providing a strong balance between speed, accuracy, and computational practicality. The technique holds substantial promise for real-world applications in logistics, transportation, aviation routing, and smart-city path planning. Future research directions include adapting the method to the Vehicle Routing Problem, the Airplane Routing Problem, dynamic routing environments, and hybrid models that combine QS-BAC with deep reinforcement learning or advanced local search techniques.

REFERENCES

1. S. M. Jasim and F. H. Ali, “Exact and local search methods for solving travelling salesman problem with practical application,” *Iraqi Journal of Science* **60**, 1138–1153 (2019).
2. A. Abdulrheem and F. H. Ali, “New heuristic techniques for solving capacitated vehicles routing problem,” *Journal of Computational Analysis & Applications* **33**, 4 (2024).
3. J. Khraibet, B. A. Kalaf, and W. Mansoor, “A new meta-heuristic algorithm for solving multi-objective single machine scheduling problems,” *Journal of Intelligent Systems* **34**, 20240373 (2025).
4. J. Bonz, “Application of a multi-objective multi traveling salesperson problem with time windows,” *Public Transport* **13**, 35–57 (2021). https://doi.org/10.1007/s12469-020-00258-6
5. W. Liu, R. Wang, T. Zhang, K. Li, W. Li, H. Ishibuchi, and X. Liao, “Hybridization of evolutionary algorithm and deep reinforcement learning for multiobjective orienteering optimization,” *IEEE Transactions on Evolutionary Computation* **27**, 1260–1274 (2022).
6. O. Dib, “Novel hybrid evolutionary algorithm for bi-objective optimization problems,” *Scientific Reports* **13**, 4267 (2023).
7. S. Zhao and S. Gu, “A deep reinforcement learning algorithm framework for solving multi-objective traveling salesman problem based on feature transformation,” *Neural Networks* **176**, 106359 (2024).
8. L. Y. Gao, R. Wang, Z. H. Jia, and C. Liu, “Multi-objective optimization for traveling salesman problem: A deep reinforcement learning algorithm via transfer learning,” *IEEE Transactions on Artificial Intelligence* (2024).
9. J. Perera, S. H. Liu, M. Mernik, M. Črepinšek, and M. Ravber, “A graph pointer network-based multi-objective deep reinforcement learning algorithm for solving the traveling salesman problem,” *Mathematics* **11**, 437 (2023).
10. B. S. Kaml and M. S. Ibrahim, “Solving the multi-objective travelling salesman problem with real data application,” *Al-Nahrain Journal of Science* **21**, 146–161 (2018).
11. N. M. Neamah and B. A. Kalaf, “Solving multi-criteria scheduling: total completion time, late work, and maximum earliness,” *Iraqi Journal of Science* **65**, 2724–2735 (2024).
12. B. A. Kalaf, G. J. Mohammed, and M. D. Salman, “A new hybrid meta-heuristics algorithms to solve APP problems,” *Journal of Physics: Conference Series* **1897**, 012011 (2021).
13. B. A. Kalaf, R. A. Bakar, L. L. Soon, M. B. Monsi, A. J. K. Bakheet, and I. T. Abbas, “A modified fuzzy multi-objective linear programming to solve aggregate production planning,” *International Journal of Pure and Applied Mathematics* **104**, 339–352 (2015).
14. T. J. Khraibet, B. A. Kalaf, and E. Rehman, “Solving multi-objective machine scheduling problem using the Meerkat Clan Algorithm,” *Journal of Economics and Administrative Sciences* **31**, 158–169 (2025).
15. M. Elyasi, B. Altan, A. Ekici, O. Ö. Özener, İ. Yanıkoğlu, and A. Dolgui, “Production planning with flexible manufacturing systems under demand uncertainty,” *International Journal of Production Research* **26**, 157–170 (2024).

16. D. R. Singh, M. K. Singh, T. Singh, and R. Prasad, “Genetic algorithm for solving multiple traveling salesmen problem using a new crossover and population generation,” *Computación y Sistemas* **22**, 491–503 (2018).
17. M. Veenstra, K. J. Roodbergen, I. F. Vis, and L. C. Coelho, “The pickup and delivery traveling salesman problem with handling costs,” *European Journal of Operational Research* **257**, 118–132 (2017).
18. Abdul-Zahra, I. T. Abbas, B. A. Kalaf, L. W. June, and M. B. Monsi, “The role of dynamic programming in the distribution of investment allocations between production lines with an application,” *International Journal of Pure and Applied Mathematics* **106**, 365–380 (2016).
19. R. Abbas and M. N. Ghayyib, “Using sensitivity analysis in linear programming with practical physical applications,” *Iraqi Journal of Science* **65**, 907–922 (2024).