

A comparison of meta-heuristic and hyper-heuristic algorithms in solving an urban transit routing problems

Ahmad Muklason, Shof Rijal Ahlan Robbani, Edwin Riksakomara, I Gusti Agung Premananda

Department of Information Systems, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

Article Info

Article history:

Received Jan 24, 2023

Revised Nov 15, 2023

Accepted Dec 3, 2023

Keywords:

Hill climbing

Hyper-heuristics

Particle swarm optimization

Simulated annealing

Urban transit routing problem

ABSTRACT

Public transport is a serious problem that is difficult to solve in many countries. Public transport routing optimization problem also known as urban transit routing problem (UTRP) is time-consuming process, therefore effective approaches are urgently needed. UTRP aims to minimize cost passenger and operator from a combination of route set. UTRP can be optimized with heuristics, meta-heuristics, and hyper-heuristics methods. In several previous studies, UTRP can be optimized with any meta-heuristics and hyper-heuristics methods. In this study we compare the performance of meta-heuristic methods, i.e. hill-climbing, simulated annealing, and hyper-heuristics method based on modified particle swarm optimization algorithm. The experimental results showed that the proposed methods could solve UTRP effectively. Regarding their performance, the results show that despite the generality of hyper-heuristics, their performance are competitive. More specifically, hyper-heuristics method is the best method compared to the other two methods in each dataset. In addition, compared to prior studies results, the proposed hyper-heuristics could outperform them in term of cost passenger of small dataset Mandl. The main contribution of this paper is that to best of our knowledge, it is the first study comparing the performance of meta-heuristics and hyper-heuristics approaches over UTRP.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Ahmad Muklason

Department of Information Systems, Institut Teknologi Sepuluh Nopember

St. Raya ITS, Kampus ITS Sukolilo Surabaya 60111, East Java, Indonesia

Email: mukhlason@is.its.ac.id

1. INTRODUCTION

Public transport is a solution to overcome traffic jams in an area or regional. However, there are still many countries that has not implemented public transport properly and optimally, so that cannot be a solution for any civilian. Traffic congestion is a serious problem in the world and not easy to handle it [1]. An implementation of public transport can be improved by optimizing routes based on distance and time [2]. But, it can take a lot of time and energy, so a can be problem classified as combinatorial problems.

A problem of determining public transport routes can also be referred to urban transit routing problem (UTRP). UTRP is the part of urban transit network design problem (UTNDP) which focuses on solving problem of finding routes and scheduling public transportation. UTNDP is the part of vehicle routing problem (VRP). UTRP has a fitness function like cost passenger and operator [3]. A cost passenger point of view is said to be efficient, if the total travel time and total transfer time can be minimized. Meanwhile, cost operator point of view the number of routes, and the total number of all bus route lengths can be minimized. To be able to minimize the UTRP fitness function, a suitable combination is needed to determine a route. Therefore, an UTRP problem

can be said to be a combinatorial problem [4]. Combinatorial problems can be solved using heuristics, meta-heuristics, and hyper-heuristics methods, because they can help find an optimal objective function by providing several possible combinations of solutions [5], [6].

Hyper-heuristics method has a major difference with metaheuristics, it directly searches for solutions in the space of heuristics [7]. Hyper-heuristics method aims to find right combination of low-level heuristics (LLH) is easy to apply and produces an acceptable domain solution [8]. Metaheuristic methods has been used to solve VRP problems, including: genetic algorithm (GA), simulated annealing (SA) algorithm, tabu search (TS) algorithm, and particle swarm optimization (PSO) algorithm [9], [10]. Hyper-heuristics have also been used to solve VRP problems and resulted well-designed hyper-heuristics are useful for building and improving solutions [2], [7], [11].

In several previous studies, VRP problems have been solved using several metaheuristic methods, including: GA, PSO, and TS. VRP solution using a hybrid GA method shows that hybrid GA method has better performance than normal GA method [12]. Completion of VRP using modified PSO shows that, modified PSO method has better performance than PSO in achieving vehicle routes that focus on problems by considering complete costs which will reduce total costs and emissions [13]. Completion of VRP using the hybrid TS method results in better performance, less computation time, and very effective in finding solutions to problems [14]. VRP solution using GA and PSO methods, shows that can produces a fairly good objective value and represented that GA is better than PSO [15]. Results of previous research indicate that a modified or hybrid metaheuristic method can be a solution to obtain a route sequence that is close to optimal solution with fast processing time.

Several studies on UTRP have been carried out. One of them is to use hill-climbing (HC) and SA methods to solve an UTRP problem, both of methods can produced competed fitness function with results of previous studies [16]. Another studies, uses a new heuristic and evolutionary operators methods, and show a results proposed method can outperform previous research in terms of passenger costs and operator costs [17]. Another studies, proposed a selection hyper-heuristics methods as a tool to solve an UTRP problem. Based on this research, it is known that the sequence-based selection methods combined with the great deluge acceptance has the best performance in terms of passenger and operator costs [18]. Another studies, has been success implemented a cat swarm optimization for UTRP and produced an objective function can be compared with previous studies [19]. Another studies, successfully used imported flower pollination algorithms to solve UTRP and had more effective results in the Mandl dataset [20]. Another studies, applied differential evolution, show that an objective function is outperformed most of the results from other literatures [21]. Another studies, proposed a sequence-based selection methods combined with the great deluge acceptance (SS-GD) using dataset development and show that methods is able to improve existing route services for passengers and operators, and shows great potential to deal with real-world problems in a short time when compared to GA [22].

Based on several previous studies, an UTRP problem can be solved using meta-heuristics or hyper-heuristics methods. Therefore, in this study, an UTRP will be solved by using HC and SA as metaheuristics methods and a modified particle swarm optimization algorithm based on gravitational search interactions (MPSO-GI) with a hyper-heuristics approach as hyper-heuristics methods. These three methods are used for solving public transport problems to get a more optimal solution of problem. So that result of cost passenger and operator as objective function can find a better solution. Reason of using a MPSO algorithm is based on previous research which explains that modified PSO algorithm has better results than standard PSO for solving VRP problems [13]. MPSO-GI algorithms modifies PSO with the gravitational search algorithm approach and shows more optimal results than standard PSO [23]. MPSO-GI has a learning coefficient value based on a position of the gravitational interactions between particles which much more stable than user input value. PSO algorithm and hyper-heuristics approach has been carried out in research on solving the resource constrained project scheduling problem (RCPSP) and has a competitive solution result when compared to other methods [24]. In research on evolving dispatching rules in job shop scheduling, PSO algorithms and hyper-heuristics approaches can be applied and produce solutions that are more competitive and faster than genetic programming methods with hyper-heuristics [25].

A dataset used in this study is dataset in previous research, namely Mandl and Mumford [26]. Mandl data is a representation of bus routes in 15 cities in Switzerland. Meanwhile, Mumford dataset is a representation of the small transport network and bus routes in the city (Yubei, Brighton, and Cardiff). The results of this study will be compared with the results of previous studies using the same dataset. So it can find out the

advantages of proposed methods in solving an UTRP problem.

2. RESEARCH METHOD

In this section, the research methodology employed to tackle the UTRP is elucidated in detail. To ensure a comprehensive understanding of the approach, the methodology is divided into sub-sections, each detailing a specific phase of the research. Firstly, the dataset utilized is discussed, which comprises examples from various global locations, including Yubei in China, Brighton and Cardiff in the UK, and 15 cities in Switzerland. To analyze this dataset, the design of the algorithm is explored in stages, encompassing data conversion, initial solution formation, and the incorporation of various algorithmic techniques such as HC, SA, and hyper-heuristics using MPSO-GI. The implementation phase details the technical setup and computational resources used, ensuring replicability of the results. Lastly, the results derived from the applied algorithms will be critically analyzed and compared with the findings of previous studies in the domain. This systematic approach ensures a holistic and structured examination of the UTRP, facilitating a comprehensive exploration of potential solutions.

2.1. Dataset

The study utilizes data from a benchmark dataset previously used in research [26]. This dataset includes five examples, called Mumford0, Mumford1, Mumford2, Mumford3, and Mandl. Mumford1, Mumford2, and Mumford3 represent bus route data in Yubei, China, Brighton, and Cardiff, UK respectively. Mumford0 represents data with a small network, while Mandl represents bus routes in 15 cities in Switzerland. Table 1 further describes the specifics and features of the dataset used in the study.

Table 1. Description instance

Instances	Vertices	Edges	Routes	Vertices per route (min-max)
Mandl4	15	21	4	2 – 8
Mandl6	15	21	6	2 – 8
Mandl7	15	21	7	2 – 8
Mandl8	15	21	8	2 – 8
Mumford0	30	90	12	2 – 15
Mumford1	70	210	15	10 – 30
Mumford2	110	385	56	10 – 22
Mumford3	127	425	60	12 – 25

2.2. Design of algorithm

In this sub-section, a structured, step-by-step algorithmic approach to address the research problem is outlined. This algorithmic strategy involves several intricate steps, including data conversion, formation of the initial solution, and application of multiple optimization techniques. Each of these steps, pivotal to the success of the research, is detailed in subsequent segments to ensure a comprehensive understanding of the approach.

2.2.1. Data conversion and hard constraints

In this step, dataset used is read in .txt format. Then dataset is converted into an array by making adjustments, so that can be read by program. Adjustments are made by replacing the value "Inf" to 0 in a TravelTimes data, so that process of finding hard constraints can be easier. Furthermore, data can be obtained in the form of an array of TravelTimes and demand data. An array of data can be optimized according to desired algorithm with program that has been created. Next, search for hard constraints by searching whether points between an arrays have a value of 0.

2.2.2. Initiate solution

In this step, an initial solution is formed from selected and implemented dataset. The first step in forming an initial solution is to determine parameters of each dataset. Parameters need to be determined include number of routes, as well as MIN and MAX lengths on each route. Parameters used are in accordance with the specifications of each dataset that have been described. Furthermore, the initial solution can be obtained by determining starting point of a solution on each route by determining point, that has most connections with

other points. Next point is selected based on the hard constraints of starting point that has been obtained. The formation of an initial solution can be explained by pseudocode in Algorithm 1.

Algorithm 1: Pseudocode for initiating solution

```

Function getNewRoute (data, Min, Max, r, datasize) :
    route ← array;
    dataC ← getConnected(data);
    startNode ← start node per route;
    for i = 0 to r do
        | initializeRoute(route[i], startNode, dataC);
    end
    while Unused is not empty do
        | randRute ← pickRandomRoute();
        | resetRoute(route[randRute], startNode, dataC);
    end
    return route;

```

2.3. Calculate fitness

To proceed to the optimization stage, it is necessary to calculate the fitness value for each iteration that will be performed. Fitness function of an UTRP is cost passenger and operator. Cost passenger (C_p) obtained from the total travel time of all passengers which can be calculated as in (1):

$$C_p(R) = \frac{\sum_{i,j=1}^n d_{ij} a_{ij}(R)}{\sum_{i,j=1}^n d_{ij}} \quad (1)$$

where d_{ij} = transit demand from node x_i to node x_j , a_{ij} = shortest time from x_i to x_j , and R = route set.

Cost operational regarding a number of vehicles to ensure service quality cannot be handled without considering vehicle scheduling. Other costs operating depend on length of a transportation route. Therefore a cost operator (C_o) defined as total length of a route set as in (2):

$$C_o(R) = \sum_{a=1}^n \sum_{(i,j) \in r} t_{ij}(a) \quad (2)$$

where: a = typical route R , r = number of routes, and $t_{ij}(a)$ = length of transport link.

2.3.1. Hill climbing

In this section, the research is carried out using a results of initial solution that has been formed including, initial route and cost of passenger and operator. Furthermore, results of the initial solution are processed using a HC method to get better solution. HC method used is make-small-change as was done in previous research [9]. Pseudocode of the HC method can be seen in Algorithm 2.

Algorithm 2: Pseudocode for HC

```

Function getRouteHC () :
    rand1, rand2 ← pickRouteAndNodeRandomly();
    if route[rand1] j max then
        | while !route[rand1].contains(rand2) do
        | | route[rand1].add(rand2);
        | end
    end
    else
        | swapStartAndEndNodes(rand1);
    end
    if not isValid then
        | resetRoute(rand1);
    end
    return route;

```

The first step of HC is to determine randomly which route will be modified. After getting the selected route, then determine the possibilities that can be used. First possibility can be used, if the length of a selected

route is less than MAX parameter. As well as last possibility can be used, if the route can not run the first possibility.

After performing the possible procedures of make-small-change method a new route is obtained. Next, check whether a route meets the requirements as a route in an UTRP problem. If a route is declared valid then, a route can replace previous route and used for calculating new fitness function.

2.3.2. Simulated annealing

In this SA algorithm, the research is conducted by utilizing the results of the initial solution, which includes the initial route and the cost for passengers and operators. The results of the initial solution are then further processed using the SA method to achieve better results. The SA method is applied with the temperature set to 1000 and the cooling factor as per previous studies [16]. Additionally, the combination of the make-small-change method, as described in previous research [9], is used. The flowchart for the SA method can be seen in Figure 1.

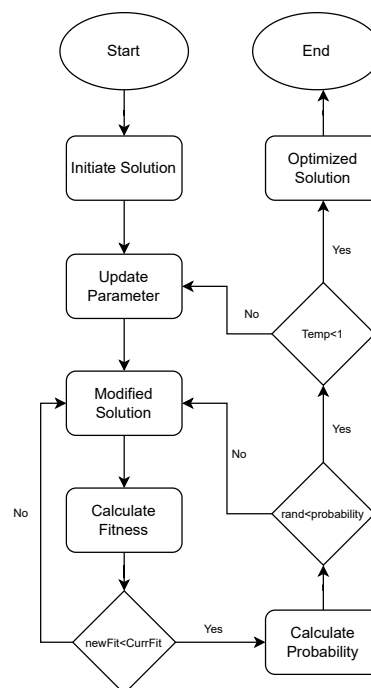


Figure 1. Flowchart SA

The first step is to determine the temperature parameter and calculate the cooling factor, which is used as an iteration for the procedure to be carried out. Next, a random route is selected to be modified. After the selected route is determined, possible modifications are evaluated. The first possibility is used if the length of the selected route is less than the MAX parameter. The last possibility is used if the first possibility cannot be applied.

After performing the possible procedures of the make-small-change method, a new route is obtained. The procedure is repeated until the temperature drops to zero. Next, the validity of the route as a solution to the UTRP problem is checked. If the route meets the requirements, it replaces the previous route and is used to calculate the fitness function.

2.3.3. Hyper-heuristic with modified particle swarm optimization

In this study, a hyper-heuristic approach that combines a modified version of MPSO-GI with simple random selection and five types of LLH is employed to solve the UTPR. The hyper-heuristic process is carried out for a pre-determined number of iterations and is constrained by a time limit. Move acceptance is used in each iteration, and the process of the hyper-heuristics can be observed in the flowchart presented in Figure 2. The UTPR optimization process using the hyper-heuristics begins with a random selection of an LLH, followed by the application of MPSO-GI until the desired number of iterations is reached.

In the move acceptance phase, MPSO-GI is used to select a solution based on the particle, and the specified number of iterations is run. The initial step of MPSO-GI involves defining the particle parameters and the number of iterations. A random route is then chosen and transformed using MPSO-GI into a new solution using the specified number of particles. The new solution is evaluated using the fitness function, and if it has a better value, it is accepted and replaces the previous solution. The procedure for implementing MPSO-GI move acceptance can be found in Algorithm 3.

Algorithm 3: Pseudocode MPSO-GI Algorithm

```

Function MPSO ( ) :
  particle  $\leftarrow$  number of solution;
  iteration, MaxIter  $\leftarrow$  number of iteration;
  generatePosition();
  generateVelocity();
  solutionPerParticle();
  while iteration;maxIter do
    newVelocity();
    newPosition();
    solutionPerPartikel();
    if bestFit < fitness then
      fitness = bestFit;
      updateRoute();
    end
  end
  return routeSR;
  
```

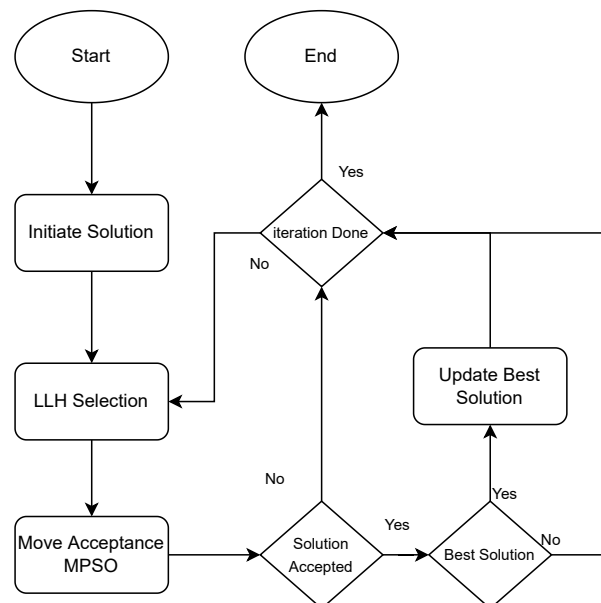


Figure 2. Flowchart hyper-heuristics

In addition, the simple random method is used to randomly select an LLH for each iteration. In this study, the number of LLH [4] used is:

- LLH0: choose a random route and position of route. Then add random value points to a position and route that has been selected.
- LLH1: randomly chooses a route and two positions in a route. Then swap two positions that have been chosen.

- LLH2: randomly chooses a route and two positions in a route. Then a first point replaces a second point, and a second point becomes a last point.
- LLH3: choose two routes randomly and a position on each route. Then enter a point on first route into second route to selected second point.
- LLH4: choose two routes randomly and one position on each route. Then change point that has been selected on each route.

2.4. Implementation

The algorithm developed in this study was implemented using the Java programming language and the eclipse integrated development environment (IDE). The computer system used for the implementation had the following specifications: an Intel i5-8250U processor with a clock speed of 1.60 GHz, 8129 MB of RAM, and the Windows 10 Home operating system. The program was executed on each dataset for 200 iterations using the HC algorithm. In the case of the MPSO-GI algorithm, the program was also run for 200 iterations, with a time constraint of 10 minutes to ensure consistency in running time with the HC algorithm. The SA algorithm was configured to produce a running time that is comparable to the other algorithms. To ensure the robustness of the results, all algorithms were run 10 times on each dataset, and the best result was selected for the analysis phase.

2.5. Analysis of results

The performance of the algorithm will be compared with the methods and outcomes of two prior studies. The first study, by Mumford [17], developed the new heuristic and evolutionary operators (NH-EO) algorithm. The second study, by Ahmed *et al.* [18], developed the sequence selection based great deluge (SS-GD) algorithm. The analysis and comparison of the results will provide insight into the effectiveness of the implemented algorithm in comparison to the previously developed ones.

3. RESULTS AND DISCUSSIONS

The results section discusses the outcome of the experiments conducted on the UTRP problem using three algorithms. The algorithms used were HC and SA, which employ the metaheuristic approach, and MPSO-GI, which utilizes the hyper-heuristic approach. In the HC method, the parameters used were 200 iterations and 10 trials, while in the SA method, the temperature was set to 1000 and the cooling factor to $0 < c < 1$. MPSO-GI used 5 particles, 200 iterations, and 10 trials. The best results from the tests will be compared to analyze the three algorithms.

3.1. Comparison result of MPSO-GI, simulated annealing, and hill climbing

The experiments compared three distinct algorithms: HC, SA, and MPSO-GI. The results, as depicted in Table 2, show notable differences in their performance across various datasets. The MPSO-GI algorithm, underpinned by the hyper-heuristic approach, generally displayed superiority. This dominance was evident in both the "best" and "average" results across the majority of datasets. However, there were exceptions. In the passenger cost (Cp) aspect, the "best" results from the HC algorithm surpassed MPSO-GI for the Mumford1 and Mumford2 datasets. Additionally, in the operation cost (Co) aspect, the "best" result of the SA outperformed MPSO-GI on the Mumford2 dataset.

When comparing HC and SA, HC typically exhibited better performance. This was particularly evident in the passenger cost aspect, where HC's "best" and "average" results were superior in most datasets, with the exception of Mumford0. For operation costs, HC's "best" results were better on the Mandl4, Mandl6, and Mandl8 datasets.

The standout performance of MPSO-GI can be ascribed to two primary factors. Firstly, MPSO-GI is a population-based algorithm, contrasting with SA and HC, which are local search algorithms. Such population-based algorithms generally possess a heightened capability for solution exploration and evasion of local optima. The second factor is MPSO-GI's hyper-heuristic approach, which incorporates various LLLHs, including the 5 types of LLHs. This not only magnifies the diversity of the search but often leads to enhanced solutions.

In a direct comparison of SA and HC, SA appeared less adept at exploiting solutions. This observation was substantiated by the "average" results, where SA often lagged behind HC. However, in more extensive datasets like Mumford0, Mumford1, and Mumford2, especially in the operation cost aspect, HC, which typ-

ically accepts superior solutions, seemed more susceptible to local optima. As a result, in these datasets, its solutions were occasionally outpaced by those of the SA algorithm.

Table 2. Comparison results of proposed method

Fitness function	Methods	Statistics	Dataset						
			Mandl4	Mandl6	Mandl7	Mandl8	Mumford0	Mumford1	Mumford2
Cp	HC	Best	18	15	16	10	15	34	70
		Average	29.95	26.3	27.9	24.75	35.9	52.7	81.1
		Worst	44	45	43	41	50	63	89
	SA	Std Dev	8.20	8.77	7.18	8.09	8.93	10.71	5.79
		Best	15	27	19	16	19	54	76
		Average	23.7	29.3	29.7	18.3	40.3	66	83.2
	MPSO-GI	Worst	35	43	44	38	66	73	89
		Std Dev	5.3	5.3	9.1	9.3	10.5	9.5	6.8
		Best	8	9	8	7	17	45	72
		Average	15	16.35	17.25	19.65	30	60.35	79.35
		Worst	30	24	35	45	39	68	83
		Std Dev	4.22	3.5	9.31	9.91	6.85	8.31	3.47
Co	HC	Best	53	90	107	113	117	1074	3624
		Average	73.7	108.55	131.05	151.9	641.95	1106.55	3791.25
		Worst	101	157	172	195	1158	1158	3948
	SA	Std Dev	12.63	16.13	16.02	24.53	497.45	33.99	153.16
		Best	69	99	107	135	394	1033	3867
		Average	70.3	105	128.44	180.9	410.3	1112.5	3955.66
	MPSO-GI	Worst	100	145	166	195	430	1203	4012
		Std Dev	10.3	17.3	17.2	22.32	33.21	60.58	40.05
		Best	41	53	68	61	340	903	3905
		Average	47.25	77.55	99.55	125.95	400.5	1013.3	3946.8
		Worst	61	101	145	196	440	1106	4002
		Std Dev	6.91	12.74	21.21	38.13	31.01	50.84	37.77

3.2. Comparison with previous studies

In this section, the results of the three algorithms are compared to those of two previous studies. The comparison results are presented in Table 3 and Figures 3 to 4. The results indicate that, on small datasets such as Mandl4, Mandl6, Mandl7, and Mandl8, the MPSO-GI algorithm performed better overall than the algorithms in the previous studies. The MPSO-GI algorithm produced worse solutions only on the Mandl7 and Mandl8 datasets in the operator cost aspect. On the other hand, the NH-EO and SS-GD algorithms performed significantly better on larger datasets such as Mumford0, Mumford1, and Mumford2.

The significant difference in performance, particularly on larger datasets, can be attributed to a number of factors. According to Ahmed *et al.* [18], the algorithm development involved a larger number of LLHs and a more complex LLH selection strategy, as well as a larger number of iterations run. In contrast, this study had to limit the number of iterations due to our time constraints. According to Mumford [17], a larger number of iterations were also employed, which led to better results on larger datasets.

Table 3. Comparison results with several previous studies

Dataset	Fitness	HC	SA	MPSO-GI	NH-EO	SS-GD
Mandl4	Cp	18	15	8	10.57	10.48
	Co	53	69	41	63	63
Mandl6	Cp	15	27	9	10.27	10.18
	Co	90	99	53	63	63
Mandl7	Cp	16	19	8	10.22	10.1
	Co	107	107	68	63	63
Mandl8	Cp	10	16	7	10.17	10.08
	Co	113	135	61	63	63
Mumford0	Cp	15	19	17	16.05	14.09
	Co	117	394	340	111	94
Mumford1	Cp	34	54	45	24.79	21.69
	Co	1074	1033	903	568	403
Mumford2	Cp	70	76	72	28.65	25.19
	Co	3624	3867	3905	2244	1330

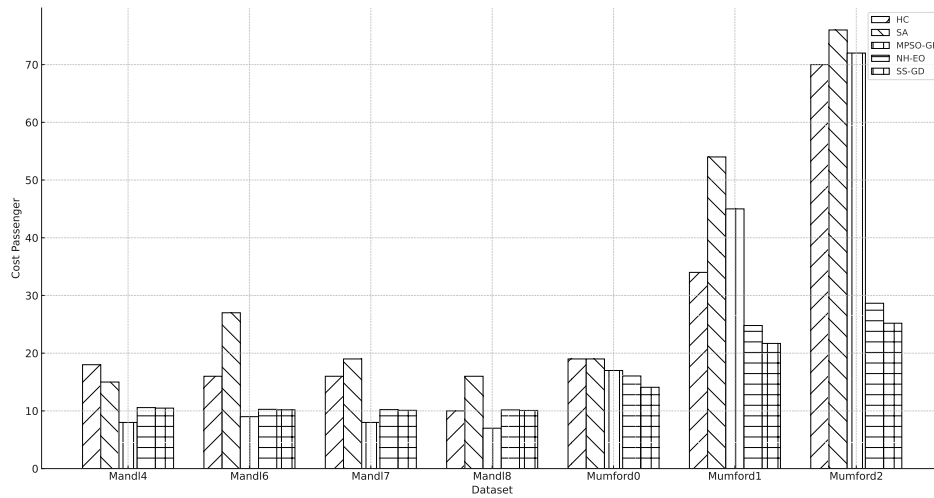


Figure 3. Comparison results of cost passenger

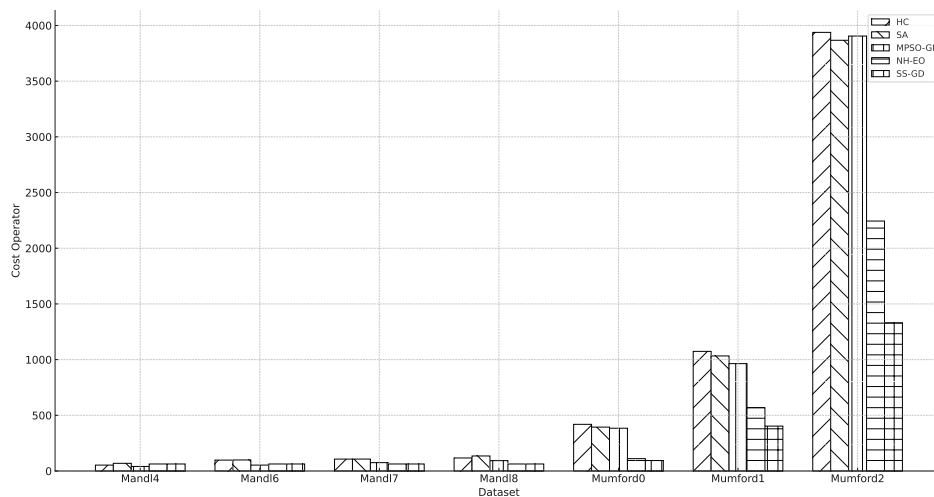


Figure 4. Comparison results of cost operator

4. CONCLUSION

In this study, a comprehensive comparison of algorithms tailored for the UTRP was conducted, with a distinct focus on the novel hyper-heuristic approach via MPSO-GI. Evaluations were benchmarked against datasets, with a particular emphasis on smaller datasets which posed unique challenges and opportunities. Empirical findings revealed that the newly introduced MPSO-GI algorithm, with its hyper-heuristic approach, consistently outperformed traditional metaheuristic techniques such as SA and HC. It's noteworthy to mention the significant influence of iteration count on the algorithmic outcomes, an aspect that warrants further exploration. Looking ahead, our research trajectory will encompass a broader analysis involving larger datasets to test the scalability of the MPSO-GI algorithm. Additionally, delving deeper into performance determinants across varying algorithms and synergizing with other heuristic methodologies for the UTRP will be pivotal components of our forthcoming investigations.




REFERENCES

- [1] A. Koźlak and D. Wach, "Causes of traffic congestion in urban areas. Case of Poland," *SHS Web of Conferences*, vol. 57, 2018, doi: 10.1051/shsconf/20185701019.





- [2] D. A. Rodriguez, P. P. Oteiza, and N. B. Brignole, "An urban transportation problem solved by parallel programming with hyper-heuristics," *Engineering Optimization*, vol. 51, no. 11, pp. 1965–1979, 2019, doi: 10.1080/0305215X.2018.1560435.
- [3] L. Fan, C. L. Mumford, and D. Evans, "A simple multi-objective optimization algorithm for the urban transit routing problem," in *2009 IEEE Congress on Evolutionary Computation*, 2009, pp. 1–7, doi: 10.1109/CEC.2009.4982923.
- [4] J. F. Cordeau, G. Laporte, M. W. P. Savelsbergh, and D. Vigo, "Chapter 6 vehicle routing," *Handbooks in Operations Research and Management Science*, vol. 14, pp. 367–428, 2007, doi: 10.1016/S0927-0507(06)14006-2.
- [5] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003, doi: 10.1145/937503.937505.
- [6] F. Peres and M. Castelli, "Combinatorial optimization problems and metaheuristics: review, challenges, design, and development," *Applied Sciences*, vol. 11, no. 14, 2021, doi: 10.3390/app11146449.
- [7] A. Tarhini, K. Danach, and A. Harfouche, "Swarm intelligence-based hyper-heuristic for the vehicle routing problem with prioritized customers," *Annals of Operations Research*, vol. 308, no. 1–2, pp. 549–570, 2022, doi: 10.1007/s10479-020-03625-5.
- [8] E. K. Burke et al., "Hyper-heuristics: a survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013, doi: 10.1057/jors.2013.71.
- [9] M. Blocho, "Heuristics, metaheuristics, and hyperheuristics for rich vehicle routing problems," in *Smart Delivery Systems*, Elsevier, 2020, pp. 101–156, doi: 10.1016/B978-0-12-815715-2.00009-9.
- [10] R. Elshaer and H. Awad, "A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants," *Computers and Industrial Engineering*, vol. 140, 2020, doi: 10.1016/j.cie.2019.106242.
- [11] W. Qin, Z. Zhuang, Z. Huang, and H. Huang, "A novel reinforcement learning-based hyper-heuristic for heterogeneous vehicle routing problem," *Computers and Industrial Engineering*, vol. 156, 2021, doi: 10.1016/j.cie.2021.107252.
- [12] M. Rabbani, P. Pourreza, H. F. -Asl, and N. Nouri, "A hybrid genetic algorithm for multi-depot vehicle routing problem with considering time window repair and pick-up," *Journal of Modelling in Management*, vol. 13, no. 3, pp. 698–717, 2018, doi: 10.1108/JM2-04-2017-0046.
- [13] Y. Li, M. K. Lim, and M. L. Tseng, "A green vehicle routing model based on modified particle swarm optimization for cold chain logistics," *Industrial Management and Data Systems*, vol. 119, no. 3, pp. 473–494, 2019, doi: 10.1108/IMDS-07-2018-0314.
- [14] J. Euchi, "The vehicle routing problem with private fleet and multiple common carriers: Solution with hybrid metaheuristic algorithm," *Vehicular Communications*, vol. 9, pp. 97–108, 2017, doi: 10.1016/j.vehcom.2017.04.005.
- [15] T. Iswari and A. M. S. Asih, "Comparing genetic algorithm and particle swarm optimization for solving capacitated vehicle routing problem," *IOP Conference Series: Materials Science and Engineering*, vol. 337, no. 1, 2018, doi: 10.1088/1757-899X/337/1/012004.
- [16] L. Fan and C. L. Mumford, "A metaheuristic approach to the urban transit routing problem," *Journal of Heuristics*, vol. 16, no. 3, pp. 353–372, 2010, doi: 10.1007/s10732-008-9089-8.
- [17] C. L. Mumford, "New heuristic and evolutionary operators for the multi-objective urban transit routing problem," in *2013 IEEE Congress on Evolutionary Computation*, 2013, pp. 939–946, doi: 10.1109/CEC.2013.6557668.
- [18] L. Ahmed, C. Mumford, and A. Kheiri, "Solving urban transit route design problem using selection hyper-heuristics," *European Journal of Operational Research*, vol. 274, no. 2, pp. 545–559, 2019, doi: 10.1016/j.ejor.2018.10.022.
- [19] I. V. Katsaragakis, I. X. Tassopoulos, and G. N. Beligiannis, "Solving the urban transit routing problem using a cat swarm optimization-based algorithm," *Algorithms*, vol. 13, no. 9, 2020, doi: 10.3390/A13090223.
- [20] L. Fan, H. Chen, and Y. Gao, "An improved flower pollination algorithm to the urban transit routing problem," *Soft Computing*, vol. 24, no. 7, pp. 5043–5052, 2020, doi: 10.1007/s00500-019-04253-3.
- [21] A. T. Buba and L. S. Lee, "Differential evolution for urban transit routing problem," *Journal of Computer and Communications*, vol. 04, no. 14, pp. 11–25, 2016, doi: 10.4236/jcc.2016.414002.
- [22] L. Ahmed, P. Heyken-Soares, C. Mumford, and Y. Mao, "Optimising bus routes with fixed terminal nodes: comparing hyper-heuristics with NSGAI on realistic transportation networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 1102–1110, doi: 10.1145/3321707.3321867.
- [23] M. Spichakova, "Modified particle swarm optimization algorithm based on gravitational field interactions," *Proceedings of the Estonian Academy of Sciences*, vol. 65, no. 1, 2016, doi: 10.3176/proc.2016.1.01.
- [24] G. Koulinas, L. Kotsikas, and K. Anagnostopoulos, "A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem," *Information Sciences*, vol. 277, pp. 680–693, 2014, doi: 10.1016/j.ins.2014.02.155.
- [25] S. Nguyen and M. Zhang, "A PSO-based hyper-heuristic for evolving dispatching rules in job shop scheduling," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 882–889, doi: 10.1109/CEC.2017.7969402.
- [26] C. Mumford, "Research on the urban transit routing problem (bus routing)," *Structure*, vol. 2, 2017.

BIOGRAPHIES OF AUTHORS







Ahmad Muklason    Assistant Professor at Data Engineering and Business Intelligence Lab., Department of Information Systems, Faculty of Intelligent Electrical, Electronic Engineering and Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. Received his doctoral degree from School of Computer Sciences, The University of Nottingham, UK, in 2017; Master of Science degree of Computer and Information Sciences Department, Universiti Teknologi Petronas, Malaysia, in 2009; and Bachelor of Computer Science from Institut Teknologi Sepuluh Nopember, Surabaya in 2006. He can be contacted at email: mukhlason@is.its.ac.id.







Shof Rijal Ahlan Robbani     IT Officer at Application Management & Operation Division, PT Bank Rakyat Indonesia. He received Master of Computer Science in Information Systems Degree from the Department of Information Systems, Faculty of Intelligent Electrical, Electronic Engineering and Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. Graduated from Information Systems Department, University of Airlangga, Surabaya Indonesia. He can be contacted at email: robbirobbani@gmail.com.



Edwin Riksakomara     Assistant Professor at Data Engineering and Business Intelligence Lab., Department of Information Systems, Faculty of Intelligent Electrical, Electronic Engineering and Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. Received his Master of Engineering degree and Bachelor of Computer Science from Institut Teknologi Sepuluh Nopember, Surabaya Indonesia. He can be contacted at email: erk@is.its.ac.id.



I Gusti Agung Premananda     received his master degree from Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia in 2021 in information systems; and Bachelor of Computer Science from Institut Teknologi Sepuluh Nopember, Surabaya in 2019. Presently he is taking the doctoral program in Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia in 2021 in information systems. He can be contacted at email: igustiagungpremananda@gmail.com.