UNIVERSITY OF EDINBURGH
Business School

2022-23

CMSE11448 Soft Computing

Individual Report

B223193

Word count: 2711

A Comparative Analysis of Genetic Algorithms and Hybrid Particle Swarm Optimization for Solving the Single Allocation p-Hub Location Problem with Flow-Dependent Cost

Abstract: The study addresses the uncapacitated Single Allocation p-Hub Location Problem with flow-dependent cost (SApHLP), with a predetermined number of hubs to open as an input parameter. The objective is to minimize transportation costs, determined by both distance and flow magnitude between nodes. Metaheuristic algorithms were coded to solve this problem: Genetic Algorithms (GA) and Hybrid Particle Swarm Optimization (HPSO). A comparison of these two techniques is presented, specifically for solving the Hub Location Problem (HLP) with a flow-dependent discount factor. The study reveals that the performance of both algorithms improves by increasing the number of hubs for CAB 25 and RGP100 datasets, with a trade-off of increased computational time. Both GA and HPSO exhibit high performance in conducted tests with no significant performance differences, but GA is observed to be more efficient in solving HLPs with relatively fewer hub choices than HPSO.
Keywords: Hub Location Problem, Flow-Dependent Cost, GA, HPSO

Introduction

The Hub Location Problem (HLP) represents a prevailing network design conundrum in Operations Research, which entails ascertaining the optimal location of hub facilities, the allotment of demand to these hubs, and the routing of flow between origin-destination pairs through these hubs, all at a minimum cost. Hubs serve as pivotal nodes that consolidate and redistribute the flow within the network. Through the use of hubs, the overall transportation cost can be decreased owing to the economies of scale. However, hub location problems also entail trade-offs between fixed costs of opening hubs, variable costs of routing flow through hubs, and service quality metrics such as travel time or distance.

A widely held assumption in hub location problems is the presence of a complete inter-hub network, signifying that each pair of hubs is connected by a direct link. Another common assumption is the absence of a direct path between non-hub nodes, which denotes that all flow must traverse through at least one hub. Based on these assumptions, hub location problems can be classified into single-allocation or multiple-allocation problems. In single allocation (SA) problems, each node is assigned to precisely one hub, whereas in multiple allocation (MA) problems, each node can utilize more than one hub for its incoming and outgoing flow.

In this assignment, I concentrate on the (uncapacitated) Single Allocation p-Hub Location Problem with flow-dependent cost (SApHLP), where the predetermined number of hubs to open is given as an input parameter. SApHLP represents a formidable problem that holds noteworthy real-world applications in transportation, logistics, and supply chain management. I have elected to opt for the two standard hub location datasets (i.e., CAB and TR) drawn from real-world data along with larger Randomly Generated Problems (RGPs). The objective of this problem is to minimize the overall transportation cost, which hinges on both the distance and

the magnitude of flow between nodes. We will present two solution techniques and algorithms that are coded using metaheuristics.

Literature Review

Goldman (1969) published his inaugural paper addressing the problem of network hub location. Morton O'Kelly (1986) originally attempted to formulate the mathematical problem of hub location, known as the quadratic IP approach and developed two simple heuristic algorithms to solve the problem of locating 2-4 hubs in a maximum of 25 potential locations. Despite the dominant use of discrete approaches in HLP research, O'Kelly (1987) proposed a discrete model based on the p-median problem, which involved adjusting the existing p-median constraints and developing an objective function that considered the interaction effects between different facilities. However, these early creative ideas were subject to the multi-facility location problem, which assumed that there were flows between different facilities. While these early HLP models may seem straightforward now, single allocation hub location models with a complete inter-hub network continue to present challenging research problems, with further developments in subsequent literature, including fixed cost models (O'Kelly, 1992) and approach adapted using various heuristics, such as genetic algorithms and tabu search. These various extensions, accumulated over the years, led to the acceptance of hub location models as a critical but specific problem in facility location theories.

Since HLP is NP-hard, solving it optimally for large instances using exact methods is challenging. Therefore, heuristic methods, such as Genetic Algorithms (GA) and Particle Swarm Optimization (PSO), have been proposed to solve HLP.

GA is a metaheuristic inspired by the principles of natural selection and genetics. In GA, a population of candidate solutions is randomly generated, and each solution is represented as a chromosome or a string of genes. The fitness of each chromosome is evaluated based on its objective function value, which represents the total transportation cost. GA uses three genetic operators, selection, crossover, and mutation, to generate new offspring solutions from the parent solutions in each generation. The best solution found in the population is returned as the final solution. However, other simple meta-heuristics, such as tabu search (Silva and Cunha, 2009), have outperformed the overall performance of published GA works in solving hub location problems. A hybrid approach based on Tabu Search and GA was proposed by Abdinnour-Helm (1998) to solve the SApHLP, where GA was applied to determine the number and location of hubs while TS was used to assign different spokes to each hub. Topcouglu *et al.* (2005) found that a GA-based approach improved the solution of CAB problems. Rieck et

al. (2014) introduced a genetic algorithm (GA) to present a location-routing model with a 3-level hierarchy in which the levels are delivery points, potential hubs, and supply points to minimize transportation costs. Cuuha (2007) introduced a heuristic based on genetic algorithms to configure hub-and-spoke networks for Brazilian truck companies. Their formulation is notable for its ability to incorporate variable scale-reduction factors for transportation costs based on the total freight volume between hubs. Additionally, an efficient local improvement procedure is applied to each individual in the population, making it stand out. Topcuoglu *et al.* (2005) proposed a robust GA-based solution for the SApHLP, demonstrating its performance and computational effectiveness by comparing its solutions with the best ones obtained from prior literature.

Particle Swarm Optimization (PSO) is another metaheuristic inspired by the behaviour of social animals like birds and fish. In PSO, a population of particles representing candidate solutions to the problem is initialized randomly, and their fitness is evaluated based on the objective function value. Originally developed by Kennedy and Eberhart (1995), PSO is a population-based metaheuristic inspired by the motion of migrating birds. PSO applies concepts of communication and collaboration within the population of particles and has undergone key refinement and further development (Banks et al., 2007; 2008) to achieve a significant performance improvement, leading to its use in diverse applicational fields (Ai and Kachitvichyanukul, 2009; Liu et al., 2012; Marinakis and Marinaki, 2010; Pedrycz et al., 2009; Sevkli and Guner, 2006; Sha and Hsu, 2008; Yang et al., 2013; Zhao et al., 2014). However, PSO is not frequently used in hub location problems. Yang et al. (2013) improved the hybrid PSO by combining genetic operators and local search algorithms, which yielded better results in terms of efficiency and quality. Recently, metaheuristic algorithms have emerged as promising techniques for solving complex optimization problems, such as HLP. Among these algorithms, PSO has demonstrated its effectiveness in solving HLP. However, it can still suffer from premature convergence and slow convergence rates. To address these limitations, researchers have proposed a hybridization approach by combining PSO with other metaheuristic algorithms. HPSO combines the strengths of PSO and other metaheuristic algorithms to improve the search capability and convergence rate of the algorithm.

This paper presents a comparative analysis of GA and HPSO for solving HLP. Additionally, unlike previous studies that concern transportation costs with fixed costs, the paper measures the inter-hub transportation cost using flow-dependent discount factors that can be situation-dependent as well.

The remainder of this essay begins with a model description and then includes discussions

of computational results and an analysis of the chosen algorithms for HLPs. The paper concludes with a summary highlighting key areas for future improvement.

Models Description

In this section, I provide an account of the entire process of constructing models for the SApHLP problem with a flow-dependent discount factor. According to Nader Azizi (2022), by comparing the values of the objective function returned by the model with a fixed discount factor and with a flow-dependent one, it shows clearly that the total transportation costs are underestimated in the solutions obtained by the model that utilizes a fixed discount factor.

To commence the project, I import data on the flow and cost between nodes as matrices denoted by 'w' and 'c,' respectively. This information serves as a foundation for subsequent matrix operations. To illustrate node allocation and hub selection, I utilize a list comprising of 'n' nodes, including hubs, to record the allocation strategy. This approach proves to be intuitive and practical since the number displayed in the list indicates the hubs uniquely, and the location/index of the number signifies which node is assigned to which hub. It is worthwhile to note that this method requires feasibility checks to confirm the index of the hub aligns with the hub number itself. Employing this strategy to present hub selection for the hub location problem proves to be more efficient than constructing a dictionary to denote whether a node is a hub or not. This solution serves as the initial solution for the next iteration of each algorithm, and such a process is repeated until the stopping criterion is satisfied, and the optimal solution is derived from the algorithm.

Firstly, I randomly generate an allocation strategy list based on predefined feasibility functions. Secondly, I define a function to calculate the cost of the given allocation policy, where I set a piecewise-linear concave cost function with three different line segments for inter-hub connection k-m (refer to Figure 1). Given that the transfer from one hub to another hub incurs a discount factor, I divide the total cost into three parts, namely collection cost, distribution cost, and inter-hub cost.

Flow (k-m)	Slope	Intercept
$0 \leq f'_{km} < 50$	1.0	0
$50 \leq f'_{km} < 100$	0.8	10000
$100 \leq f'_{km} < 200$	0.6	30000
$200 \leq f'_{km}$	0.4	70000

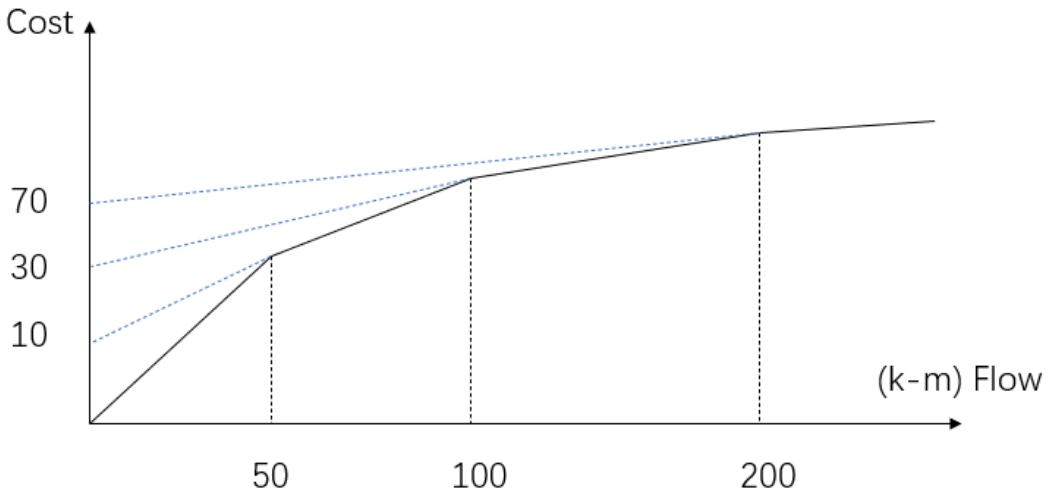

Figure 1. Piecewise-Linear Concave Cost Function on Inter-hub link k-m

After preparing all the processes required for the execution of two different algorithms (GA & PSO+GA), I begin by defining the class for these two algorithms and recording the time for each running of the algorithms. The stopping condition is the number of iterations, and after several trials, the running time is fixed at approximately 30 minutes to obtain all ten running results.

For GA, we utilize a one-point crossover operator to generate offspring solutions from parent solutions. Firstly, two feasible solutions/parents are generated as the starting point of the GA. Secondly, a randomly selected point serves as the cutting point of each parent node, combining the two parts of each parent as their offspring. We employ a bit-flip mutation operator to randomly alter some bits in the offspring solutions. However, to ensure that the offspring meets the allocation assumptions, it must pass the feasibility check to create a new population. This approach provides an overall satisfactory and explicit way to express the construction of the algorithms. The process of selection, crossover, and mutation continues

until a stopping criterion is met, such as a maximum number of generations or a satisfactory fitness value. In this study, each algorithm runs for a fixed time, depending on the size of the benchmark problems. Refer to Figure 2 for the pseudocode of GA.

Input: GAs parameters *//Population size, Operators' rates, Stopping Condition*
Output: a (near-) optimum solution(s)
Generate the initial population Pop_0
$t = 0$
repeat
  for each individual $i \in Pop_t$ **do**
    Assign fitness to individual *i*.
  end for
  Select solutions from Pop_t based on their fitness values
  Apply crossover/mutation operations on the selected solutions
  Perform replacement to obtain next generation Pop_{t+1}
  $t = t+1$
until the stopping condition is met.

Figure 2. The pseudocode of the Genetic Algorithm

The Hybrid PSO-GA algorithm employs two distinct algorithms through separate functions. In PSO, each candidate solution is represented as a vector in the search space, with each element in the vector indicating the probability of selecting a location as a hub. Initially, a population of particles is randomly initialized in the search space. An initial solution representation is generated using a 2*n matrix consisting of two rows. The first row displays n uniform distributed continuous values ranging from 0 to p*n, which aim to randomly choose p potential hubs locations by ranking all elements and selecting the first p elements. The second row generates n different locations for allocation. As the hubs are determined by the first row, allocation in the second row can only select from these p hubs, meaning that the continuous value can only change from 1 to p and take the integer parts as the choice of which hub the node should be allocated to. Feasibility prerequisites must be met for the solution evaluation process to continue. The PSO pseudocode is presented in Figure 3.

```
Start
  i = 1
  Do while i < SwarmSize
        Take a two-dimensional particle position matrix with continuous values
        Rank all elements in the first row
        Consider the first pelements in the first row as potential hub locations
        Take the integer parts of the numbers in the second row
        Evaluate the solution
        If incomplete, repair the solution
i ← i + 1
  Loop
End
```

Figure 3. Pseudocode to convert a continuous particle position into a discrete solution

To increase the diversity of the solutions generated by PSO, GA is introduced as the diversification part. The aim is to find the point closest to the best point in the very first place. The single point-constructive crossover operator is used, following the framework demonstrated in Figure 4.1. Two particles are generated according to the PSO rule, and the new particle follows the first row taken from one of the particle/parents. Different gene fragments are combined by dividing the parent's genes by a single point. The HSPSO pseudocode is presented in Figure 4.2.

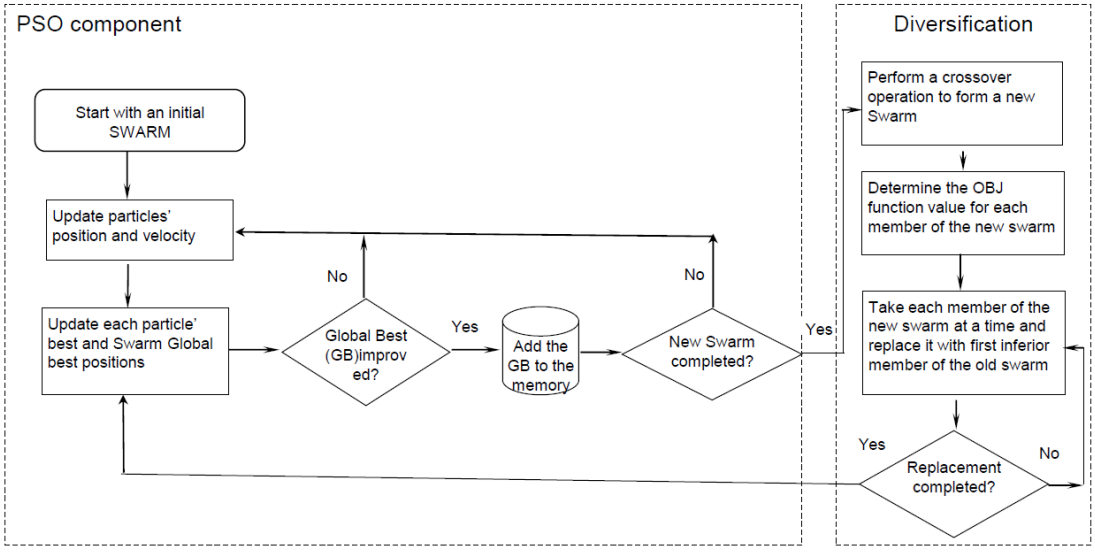

Figure 4.1. The Framework of integrating PSO&GA

```
i=1
While i < SwarmSize+1 Do
    create a template matrix for an offspring particle
    select two particles from the current swarm
    j=1
    select one of the two particles
        while j<p+1 Do
        Transfer item j from the first row in the particle matrix to the same place in the
        offspring particle matrix
        j ← j+1
        select the other particle
        End while
      Apply the classical single-point crossover operation to the second rows of the two
      particles and complete the offspring matrix
 i ← i+1
End while
```

Figure 4.2. Pseudocode of the single point-construction crossover operator for particles (matrix format) used in H-PSO

Computational results and analysis

To compare GA and HPSO for solving HLP, both algorithms were implemented using the same set of parameters and tested on a set of benchmark instances. The quality of the solutions obtained by each algorithm in terms of the objective function value and the computational time was measured. The convergence speed and robustness of the algorithms were also compared by analyzing their performance on different instances and varying the parameters.

The two algorithms were run for 8 different test problems on 4 different datasets (CAB25, TR55, TR81, and RGP100), with the number of hubs for each dataset ranging from 3 to 10 respectively. The swarm size was set according to the maximum running time for all test problems. Note that the solutions found for all the instances are not globally optimal, but the algorithms achieved global optimality when tested with the CAB10 dataset, which can be verified in the code. After running the designed algorithms 10 times using different initial solutions, a series of computational analyses were conducted, and the following results were obtained. In the computational experiments, three different piecewise-linear concave cost functions with four distinctly different slopes proposed by Klincewicz (2002) were applied.

Table 1. Computational Results: GA/HPSO

Problem	p(1)	Best Solution Configuration	Best NC(2)	Ave. NC(3)	Time(4)(sec)/# of Iterations
CAB25	3	[4, 17, 17, 4, 4, 8, 8, 8, 17, 8, 4, 8, 4, 17, 4, 17, 17, 17, 8, 4, 4, 8, 8, 4, 17]	9912996105	10977705203	0.017753816
	5	[13, 13, 17, 4, 4, 4, 13, 8, 17, 17, 11, 8, 13, 4, 13, 13, 17, 17, 13, 17, 4, 8, 8, 17, 17]	9905583868	10926855687	0.017779822

TR55	3	[4, 45, 45, 4, 4, 30, 45, 45, 4, 45, 4, 30, 30, 30, 45, 4, 30, 45, 45, 30, 30, 4, 4, 30, 45, 30, 30, 4, 45, 30, 4, 30, 30, 45, 45, 45, 45, 30, 45, 30, 4, 30, 45, 45, 45, 45, 45, 4, 30, 4, 45, 30, 45, 4, 4]	37805656964	40330716264	0.069401867
	5	[1, 1, 33, 4, 33, 45, 45, 26, 33, 4, 1, 26, 26, 4, 45, 26, 26, 4, 4, 33, 45, 1, 1, 4, 45, 26, 33, 4, 1, 1, 45, 26, 33, 1, 45, 4, 33, 45, 4, 4, 45, 26, 45, 26, 45, 26, 45, 1, 26, 45, 33, 4, 33, 4, 33]	37769979376	40907670641	0.065751113
TR81	5	[40, 16, 40, 40, 42, 6, 6, 38, 16, 40, 38, 38, 16, 40, 40, 16, 6, 38, 40, 42, 16, 16, 6, 16, 42, 40, 40, 42, 38, 38, 40, 40, 38, 16, 16, 16, 38, 38, 6, 40, 38, 42, 42, 16, 40, 6, 16, 40, 38, 42, 16, 40, 38, 16, 42, 38, 16, 38, 42, 16, 38, 42, 38, 16, 42, 38, 40, 42, 42, 42, 6, 6, 6, 38, 6, 38, 6, 6, 16, 6, 40]	73644968921	76699018719	0.195329586
	7	[6, 80, 11, 6, 34, 6, 80, 67, 18, 67, 11, 67, 6, 18, 60, 11, 67, 18, 34, 18, 60, 34, 18, 34, 60, 6, 18, 11, 18, 18, 34, 6, 34, 34, 34, 34, 6, 11, 18, 6, 6, 34, 6, 18, 18, 34, 80, 60, 60, 11, 18, 60, 80, 67, 18, 60, 6, 60, 11, 60, 11, 6, 67, 6, 18, 60, 67, 67, 18, 34, 18, 11, 34, 6, 18, 80, 11, 67, 18, 80, 11]	76643880805	78703544247	0.202210661
RGP100	7	[1, 11, 48, 11, 1, 45, 31, 58, 48, 45, 11, 11, 11, 58, 58, 23, 48, 31, 48, 11, 48, 31, 23, 45, 23, 48, 1, 48, 48, 1, 31, 45, 45, 11, 58, 11, 45, 11, 11, 1, 48, 48, 58, 1, 45, 11, 48, 48, 48, 31, 31, 45, 58, 48, 11, 1, 31, 58, 58, 23, 11, 31, 1, 48, 11, 1, 23, 23, 1, 1, 58, 1, 1, 11, 23, 31, 23, 31, 31, 1, 58, 58, 45, 1, 1, 1, 1, 48, 45, 31, 1, 1, 11, 31, 58, 31, 23, 48, 45, 23]	157058830764	159228733532	0.331575636
	10	[1, 53, 72, 36, 32, 86, 32, 42, 42, 45, 1, 1, 53, 32, 32, 86, 42, 53, 1, 53, 42, 86, 80, 72, 86, 45, 72, 53, 1, 80, 36, 32, 45, 1, 32, 36, 1, 53, 1, 36, 41, 42, 72, 42, 45, 45, 72, 45, 86, 53, 72, 53, 53, 41, 72, 32, 45, 45, 86, 1, 80, 45, 53, 72, 42, 41, 45, 41, 42, 72, 80, 72, 72, 1, 72, 1, 86, 36, 1, 80, 32, 36, 86, 42, 72, 86, 32, 32, 72, 86, 41, 36, 80, 32, 86, 45, 53, 80, 53, 80]	156197938323	158745079924	0.367760685

(1)Number of hubs;(2) Network Cost pertains to the best solution optioned by your algorithm; (2) Average NC over 10 runs;(4) Computational time/iteration number etc

Table 1 displays the optimal hub location and demand allocations, including the best solution achieved over 10 runs and the corresponding time when the optimal solution was found within the constrained computational time of 30 minutes. For instance, the selection of hubs [4, 8, 17] is denoted by [4, 17, 17, 4, 4, 8, 8, 8, 17, 8, 4, 8, 4, 17, 4, 17, 17, 17, 8, 4, 4, 8, 8, 4, 17] for the first problem instance. The computational time for each iteration ranges from 0.02s to 0.37s, and larger problems generally require more time per iteration. The algorithms implemented in this study achieved local optimality for all problem instances in less than 1 second, which demonstrates their high efficiency and capability of handling large instances under resource limitations. Furthermore, for each dataset, the choice of the number of hubs did

not significantly affect the best or average network cost or computational time. However, the combined performance of the two algorithms improved with more hubs for CAB 25 and RGP100 datasets, albeit at the expense of longer computational time. This rule did not apply to TR81, where fewer hubs yielded better quality and effectiveness.

Table 2 illustrates that both algorithms performed well and did not have significant performance differences in the tests conducted. For CAB 25 with 3 hubs, GA had better performance than HPSO, while the opposite was observed for 5 hubs. A similar pattern emerged for RGP100 regarding the performance of 7 and 10 hubs. However, for TR55, GA outperformed HPSO regardless of the number of hubs. Generally, GA was more effective in solving HLPs with relatively fewer hub choices than HPSO. Moreover, while GA achieved a lower network cost than HPSO for TR81 with 7 hubs, its average network cost was higher, indicating that the optimal point was obtained by chance and the distribution of optimal points had a large standard deviation.

Table 2. Performance Comparisons: GA vs HPSO

Problem	p(1)	GA		HPSO	
		Best NC(2)	Ave. TNC(3)	Best NC(2)	Ave. TNC(3)
CAB25	3	9912996105	10977705203	10677312169	11123382464
	5	10595603806	11005493789	9905583868	10926855687
TR55	3	37805656964	40330716264	38430790416	41030882077
	5	37769979376	40907670641	40040825706	41187985628
TR81	5	74059286089	77005790475	73644968921	76699018719
	7	76643880805	79084357780	77425659691	78703544247
RGP100	7	157058830764	159228733532	157605966746	159345700728
	10	158121382415	158874169506	156197938323	158745079924

Analyses of key parameters revealed that larger sizes and longer run times increased the likelihood of achieving global optimality, although this required more computational resources. Our experimental results demonstrated that both GA and HPSO were effective in solving HLPs and produced good-quality solutions within reasonable computational time. However, GA outperformed HPSO in terms of solution quality, especially for larger instances, while HPSO showed better convergence speed and robustness with fewer evaluations and less sensitivity to parameter selection. The choice between the two algorithms depends on the specific problem requirements and computational resources available.

Conclusion

The study presents a comparison of Genetic Algorithms (GA) and Hybrid Particle Swarm Optimization (HPSO) for solving the Hub Location Problem (HLP) with a flow-dependent

discount factor. Both algorithms exhibited good performance in finding high-quality solutions. GA produced better solutions, while HPSO showed better convergence speed and robustness. However, they are susceptible to getting stuck in local optima due to the absence of mechanisms for escaping them. The study found that increasing the number of hubs improved the performance of both algorithms at the expense of longer computation times. Future research could explore the possibility of combining both algorithms to achieve better results. Other sophisticated algorithms, such as the Efficient Hub Location with Contraction (EHLC) method (Sebastian Wandelt, 2021), have been developed to handle larger datasets or more complex problems. EHLC provides solutions with similar or better quality compared to GA and HPSO in less time.

References

Abdinnour-Helm, S. (1998), A Hybrid Heuristic for the Uncapacitated Hub Location Problem, *European Journal of Operational Research*, 106(2), 489–499.

Ai, T.J. and Kachitvichyanukul, V. (2009), A Particle Swarm Optimization for the Vehicle Routing Problem with Simultaneous Pickup and Delivery, *Computers & Operations Research*, 36(5), 1693–1702. Available at: https://doi.org/Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X).

Azizi, N. and Salhi, S. (2022), Reliable Hub-and-Spoke Systems with Multiple Capacity Levels and Flow Dependent Discount Factor, *European Journal of Operational Research*, 298(3), 834–854.

Banks, A., Vincent, J. and Anyakoha, C. (2007), A Review of Particle Swarm Optimization. Part I: Background and Development, *Natural Computing*, 6(4), 467–484.

Banks, A., Vincent, J. and Anyakoha, C. (2008), A Review of Particle Swarm Optimization. Part II: Hybridisation, Combinatorial, Multicriteria and Constrained Optimization, and Indicative Applications, *Natural Computing*, 7(1), 109–124.

Goldman, A.J. (1969), Optimal Locations for Centers in a Network, *Transportation Science*, 3(4), 352–360.

Kennedy, J. and Eberhart, R. (1995), Particle Swarm Optimization, in *Proceedings of ICNN'95 - International Conference on Neural Networks*, *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1942–1948 vol.4.

Klincewicz, J.G. (2002), Enumeration and Search Procedures for a Hub Location Problem with Economies of Scale, *Annals of Operations Research*, 110(1), 107.

Liu, Y., Wu, X. and Hao, F. (2012), A New Chance–Variance Optimization Criterion for Portfolio Selection in Uncertain Decision Systems, *Expert Systems with Applications*, 39(7), 6514–6526.

Marinakis, Y. and Marinaki, M. (2010), A Hybrid Genetic – Particle Swarm Optimization Algorithm for the Vehicle Routing Problem, *Expert Systems with Applications*, 37(2), 1446–1455.

O'Kelly, M.E. (1986), Activity Levels at Hub Facilities in Interacting Networks, *Geographical Analysis*, 18(4), 343–356.

O'kelly, M.E. (1987), A Quadratic Integer Program for the Location of Interacting Hub Facilities, *European Journal of Operational Research*, 32(3), 393–404.

O'Kelly, M.E. (1992), Hub Facility Location with Fixed Costs, *Papers in Regional Science*, 71(3), 293–306.

Pedrycz, W., Park, B.J. and Pizzi, N.J. (2009), Identifying Core Sets of Discriminatory Features Using Particle Swarm Optimization, *Expert Systems with Applications*, 36(3, Part 1), 4610–4616.

Rieck, J., Ehrenberg, C. and Zimmermann, J. (2014), Many-to-Many Location-Routing with Inter-Hub Transport and Multi-Commodity Pickup-and-Delivery, *European Journal of Operational Research*, 236(3), 863–878. Available at: https://doi.org/Vehicle Routing and Distribution Logistics.

Sevkli, M. and Guner, A.R. (2006), A Continuous Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem, in M. Dorigo, L.M. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle (eds) *Ant Colony Optimization and Swarm Intelligence*, Berlin, Heidelberg: Springer (Lecture Notes in Computer Science), 316–323. Available at: https://doi.org/Lecture Notes in Computer Science.

Sha, D.Y. and Hsu, C.-Y. (2008), A New Particle Swarm Optimization for the Open Shop Scheduling Problem, *Computers & Operations Research*, 35(10), 3243–3261. Available at: https://doi.org/Part Special Issue: Search-based Software Engineering.

Silva, M.R. and Cunha, C.B. (2009), New Simple and Efficient Heuristics for the Uncapacitated Single Allocation Hub Location Problem, *Computers & Operations Research*, 36(12), 3152–3165. Available at: https://doi.org/New developments on hub location.

Topcuoglu, H., Corut, F., Ermis, M. and Yilmaz, G. (2005), Solving the Uncapacitated Hub Location Problem Using Genetic Algorithms, *Computers & Operations Research*, 32(4), 967–984.

Wandelt, S., Dai, W., Zhang, J., Zhao, Q. and Sun, X. (2021), An Efficient and Scalable Approach to Hub Location Problems Based on Contraction, *Computers & Industrial Engineering*, 151, 106955.

Yang, K., Liu, Y. and Yang, G. (2013), An Improved Hybrid Particle Swarm Optimization Algorithm for Fuzzy P-Hub Center Problem, *Computers & Industrial Engineering*, 64(1), 133–142.

Zhao, F., Tang, J., Wang, J., and Jonrinaldi (2014), An Improved Particle Swarm Optimization

with Decline Disturbance Index (DDPSO) for Multi-Objective Job-Shop Scheduling Problem, *Computers & Operations Research*, 45, 38–50.