Multi-Objective Location-Aware Service Brokering in Multi-Cloud - A GPHH Approach with Transfer Learning

Yuheng Chen[1], Tao Shi[2(✉)], Hui Ma[1[0000−0002−6232−4436]], and Gang Chen[1[0000−0002−9597−497X]]

[1] Victoria University of Wellington, New Zealand
{chenyuhe | hui.ma | aaron.chen}@ecs.vuw.ac.nz
[2] Qingdao Agricultural University, Qingdao, China
shitao@qau.edu.cn

Abstract. With the increasing number of cloud services in multi-cloud, it has been a challenging task to choose suitable cloud services in consideration of multiple conflicting objectives. Multi-objective location-aware service brokering in multi-cloud aims to find a set of trade-off solutions that minimize both the cost and latency. To achieve this goal, existing approaches either manually design brokering heuristics or automatically generate heuristics via Genetic Programming Hyper-Heuristics (GPHH) for each problem domain from scratch. However, manually designing heuristics takes a long time and requires domain knowledge. Also, knowledge learnt from one problem domain can be helpful for solving another problem domain. To effectively and efficiently generate heuristics for any new problem domain, we propose three novel GPHH-based approaches with transfer learning to automatically generate a group of Pareto-optimal heuristics. Experimental evaluations on real-world datasets demonstrate that our proposed GPHH with transfer learning approaches can outperform existing approaches.

Keywords: Multi-objective optimization · Multi-cloud · Service brokering · Genetic programming · GPHH · Transfer learning.

1 Introduction

Service brokering in multi-cloud plays an important role in helping application providers find services in multi-cloud to deploy their applications [22]. It has become a challenging task in finding the suitable services, as there are numerous data centers for application providers to select. As brokers, they have the problem to recommend a list of candidate services from different cloud providers to meet practical requirements. On the one hand, cloud services in different regions can have significantly varied prices. Take m6g.large from Amazon Web Service (AWS) as an example, the price of the service in North Virginia is $0.077, while the price of the same service provided in Sao Paulo is $0.1224. On the other hand,

the network latency between consumers and data centers may affect the performance of cloud applications. In order to provide cloud services with low latency, cloud providers set up their data centers across different locations. Therefore, we need to consider the balance between the performance in terms of network latency and the cost of cloud services. In this paper, we study the problem of multi-objective location-aware service brokering (MOLSB) in multi-cloud that simultaneously consider both cost and performance of services.

There are many existing works on multi-cloud service brokering problems. Most of them assume that all the requests are assigned at the same time, which is static problem [8, 18, 19, 24]. A genetic algorithm (GA) approach was proposed in [19], merging the two objectives, i.e., the cost and network latency, by a weighted sum function. Other works assume that the requests are assigned dynamically once they arrived. In [4], a multi-objective genetic programming hyper-heuristic (GPHH) approach, named GPHH-MOLSB, was proposed to generate automatically designed non-dominating rules for assigning requests. The rules designed by GPHH-MOLSB are effective in solving problem instances of the same problem domain. However, new rules need to be evolved by GPHH-MOLSB from scratch for new problem domains, ignoring the fact that the knowledge learned from other problem domains can be used. Therefore, it is desirable to design effective transfer learning methods to effectively transfer the knowledge learned from previous domains while designing new rules. Some transfer learning methods have been proposed for single-objective combinatorial optimisation problems, e.g., [1] for arc routing problems, [9] for workflow scheduling problems. However, to the best of our knowledge, there is no transfer learning method proposed for GPHH in solving multi-objective combinatorial optimisation problems, in particular for the MOLSB probem.

The aim of this paper is to develop new GPHH approaches with transfer learning techniques that can automatically design heuristics for the MOLSB problem. To achieve this, we will

1. Propose three multi-objective GPHH approaches with different transfer learning techniques. Each of the approach can generate a Pareto Front of heuristics;
2. Evaluate the proposed approaches and compare their performance with an existing GPHH based approach, using datasets collected from the real world;
3. Analyse the results of the experiments in terms of Hypervolume (HV) and Inverted Generational Distance (IGD).

The remaining sections of this paper are organised as follows. In Section 2, we review the related existing work in this area. The problem is described and modeled in section 3. In section 4, we present the GPHH approaches with transfer learning techniques to solve the dynamic MOLSB problem. Section 5 discusses the design of the experiments and analyses the experimental results. Section 6 concludes the paper.

2 Related Works

Various heuristic and meta-heuristic approaches have been proposed to solve the static MOLSB problem. In [12], a greedy algorithm was proposed to solve the problem. Since the greedy algorithm only allows requests arriving at the same time, it will get stuck in local optima. To improve the algorithm proposed in [12], a genetic algorithm (GA) approach was proposed to optimise the static MOLSB problem in [19]. These approaches assume all the application deployment requests are known and arrive at the same time.

Hyper-heuristic methods are used to automatically generate heuristic rules to solve dynamic combinatorial optimisation problems. Genetic programming (GP), as an evolutionary computation technique, aims to evolve and generate rules to solve a specific task. In each GP generation, genetic operators, including selection, crossover and mutation, are applied to evolve GP rules [16]. Through many generations of evolution, a final rule will be selected by the algorithm. Hyper-heuristic algorithms aim to explore in heuristic space, rather than the solution space [3]. For dynamic optimisation problem, GPHH has been applied in many areas [28, 29] for single-objective problems. To solve dynamic multi-objective multi-cloud brokering problem, [4] proposed a GPHH approach, named GPHH-MOLSB, with a newly designed terminal set, to automatically generate a set of trade-off heuristics for users to choose according to their QoS preferences. The rules generated by GPHH-MOLSB significantly outperformed the heuristics that were human-designed.

Transfer learning is a technique that can solve novel tasks by using the knowledge learned from previous tasks [6]. With the help of transfer learning techniques, evolutionary algorithms may obtain a better initial performance, reduce computational time and achieve better results in the target domain, compared to the algorithms without transfer learning [9,26]. In [6], the authors listed three main transfer approaches in genetic programming in solving symbolic regression problems, including transferring the full tree from the final generation of the source domain, transferring selected sub-trees with high fitness from the final generation of the source domain and transferring the best individuals at each generation of evolution performed in the source domain. In [9], a new transfer learning approach of GPHH was proposed for dynamic multi-workflow scheduling problem. In order to reduce the computational cost on target domain, a randomly initialized population is trained on a simpler source domain for a predefined number of generations. Then the population is evolved further in the tougher target domain for another predefined number of generations. The transfer learning approach saves the overall computation time by reducing the number of generations to be performed in the target domain. In [1], the authors proposed a novel genetic programming approach with knowledge transferring to solve the uncertain capacitated arc routing problem. The author compared the proposed method with several existing transfer learning approach, including DDGP [2], FullTree [6], GATL [15], SubTree [6] and TLGPC [13].

Most of the existing transfer learning approaches for GPHH as summarized above were proposed for solving single objective problems. In multi-objective

problems, existing approaches are not applicable or effective. For example, the TLGPC approach selects the subtrees of the individuals which are better than the mean value of the fitness in the final generation. This approach cannot be applied to multi-objective problems since the final generation provides a Pareto Front of individuals, and each of them is non-dominated. To satisfy the requirement for application deployment with different QoS preferences, we need to generate a set of trade-off solutions for users to choose. Therefore, we need to design GPHH approaches with transfer learning that can be applied to solve multi-objective problems.

3 Problem Definition

The MOLSB problem aims to allocate dynamically arriving applications to suitable cloud resources, i.e., VM instances in data centers, so as to minimize both the application deployment cost and the network latency. In this section, we present a formal model of the MOLSB problem. Constrains and assumptions of this problem are also formulated. The key notations to be used for problem definition are listed in Table 1.

For dynamically arriving application deployment requests S, a broker selects VMs at different locations from multiple cloud providers. Let $\mathcal{R}$ denotes the set of different region of data centres. In each region $r \in \mathcal{R}$, different types of VM instances are provided by cloud providers, denoted by V. Each VM type $v \in V$ has different price $C_{v,r}$ in different region $r \in R_v$. We use G_v and M_v to denote the capacities of CPU and memory that are provided by a VM type v. The set of all regions that provide VM type v is denoted as R_v

During a time period T (e.g., one day), a broker receives a sequence of requests. We use N to denote the total number of received requests. Each request i from user location u_i has two types of resource requirements, i.e., CPU g_i and memory m_i, and the time period t_i it will use the VM instance for. Once a new request i arrives at time T_i, the broker will assign it to an instance of VM type v in region r. Here, $L_{i,r}$ denotes the network latency between user location u_i and data center region r.

Following [12], we have the following assumptions about VM instances in this paper.

- The price of a VM instance can be affected by different service providers, different VM types, and different regions of data centers.
- The configuration of a VM instance cannot be modified if the VM instance has a request assigned to.
- Any single VM instance of type $v \in V$ can only have one request assigned to it at a time.
- Each request i must be assigned to exactly one VM instance of any type $v \in V$.

Table 1: Mathematical notations

Notation	Definition
V	Set of VM types
$\mathcal{R}$	Set of regions that multi-cloud data centers span
G_v	CPU capacity of VM type v
M_v	memory capacity of VM type v
R_v	Set of available regions of VM type v
$C_{v,r}$	The unit price of VM type v in region r
T	Time span of an application deployment
N	Total number of requests during time span T
T_i	arrival time of Request i
u_i	user location of Request i
g_i	CPU requirement of Request i
m_i	memory requirement of Request i
t_i	VM usage time for request i
$L_{i,r}$	Network latency between request i and region r
$x_{i,v,r}$	Binary variable indicating whether request i is assigned to VM type v in region r
TC	Total cost of the selected VMs
ANL	Average network latency of the selected VMs

The following constraints should be satisfied if a request i is assigned to a VM instance v.

$$\begin{aligned} g_i &\leqslant G_v, \\ m_i &\leqslant M_v. \end{aligned} \tag{1}$$

Eq. (1) implies that the VM instance must satisfy the resource requirement of the assigned request. A VM type is *capacity-feasible* to a request i, if the capacity of CPU is greater than or equal to g_i and the capacity of memory is greater than or equal to m_i.

The total cost of VM instances (TC) and the average network latency between users and data centres (ANL) can be calculated as follows:

$$\begin{aligned} TC &= \sum_{i=1}^{N} \sum_{v \in V} \sum_{r \in R_v} C_{v,r} t_i x_{i,v,r} \\ ANL &= \frac{1}{N} \sum_{i=1}^{N} \sum_{v \in V} \sum_{r \in R_v} L_{i,r} x_{i,v,r}, \end{aligned} \tag{2}$$

where $x_{i,v,r} \in \{0, 1\}$ determines whether request i is assigned to an instance of VM type $v \in V$ in region $r \in R_v$.

Therefore, for application deployment requests MOLSB aims to find best resources available from multiple cloud with two objectives, minimizing TC and

minimizing ANL, as defined in eq. (3):

$$\begin{aligned} &\min \quad TC, \\ &\min \quad ANL. \end{aligned} \tag{3}$$

4 GPHH with transfer learning for MOLSB

To effectively evolve brokering rules for MOLSB, we propose a GPHH algorithm with three different transfer learning approaches.

An overview of the GPHH with transfer learning approaches is shown in Fig. 1. The transfer learning approaches initialise a population of brokering rules from the previously trained rules in a source domain. In each generation, these rules are evolved by genetic operators, and evaluated by a set of training instances generated from the target domain. After a predefined number of iterations, a set of brokering rules are generated to solve the dynamic MOLSB problem on the target domain.

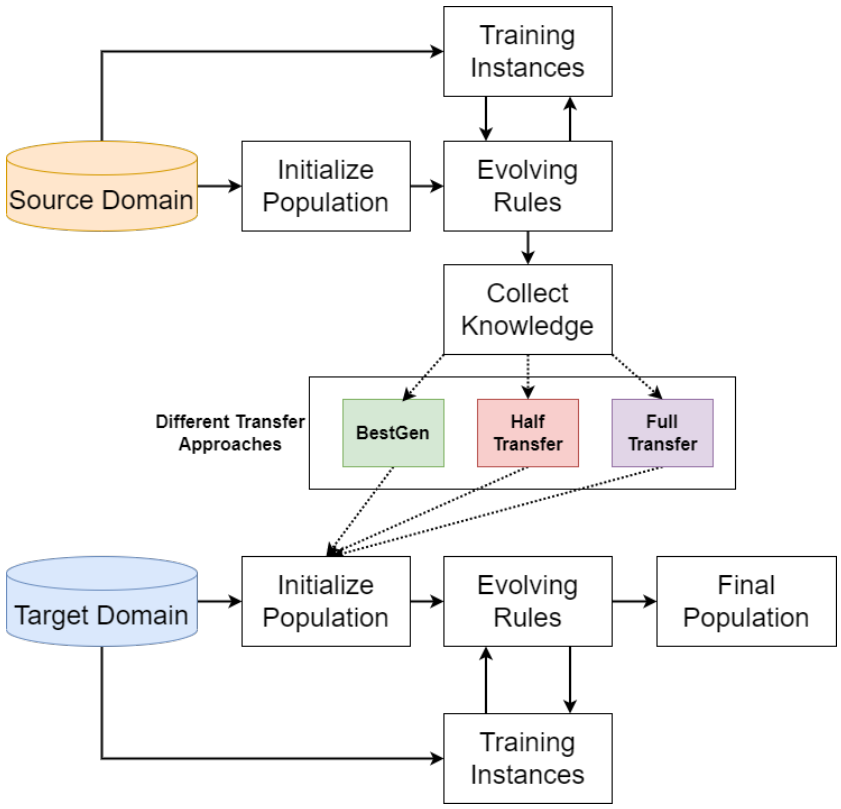


Fig. 1: The training progress of GPHH transfer learning approaches

4.1 Representation and Terminal Set

We use trees to represent the mathematical expressions of brokering rules, where the leaves are terminals and intermediate nodes are functions. To evolve brokering rules for MOLSB, we need a set of problem related features for terminal nodes as well as a set of arithmetic functions for internal nodes. We use the same set of features for terminals as in [4], since they are important features of the problem. The terminal set and function set are summarized in Table 2. Note that we use the same terminal and function set for both source and target domains.

Table 2: Terminal Set And Function Set

Terminal	Symbol	Definition
CPU	g_i	Request i's CPU requirement
Memory	m_i	Request i's memory requirement
Time	t_i	VM usage time for request i
Latency	$L_{i,r}$	The latency of request i from the data center in region r to the user
BestPrice	$min(C_{v,r})$	The minimum price of VM type v in region r that satisfies the constraints in eq. (1).
Function set	+, -, ×, protected division, maximum, minimum, cosine and sine	

An example of multi-cloud brokering rule represented as a GP tree using the terminal and function set is shown in Fig. 2.

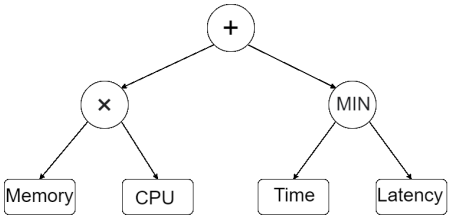
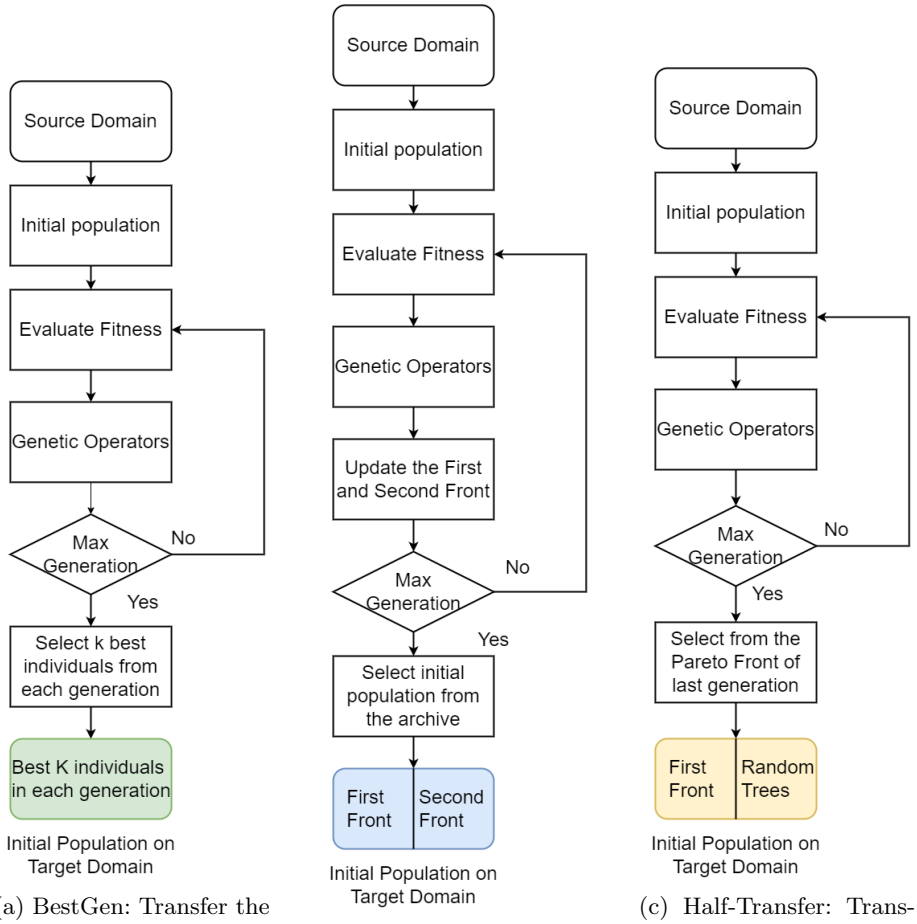


Fig. 2: An example of the tree-based representation

4.2 Transfer Approaches

To investigate using transfer learning in GPHH for solving the Multi-Objective Location-aware Service Brokering (MOLSB) problem, we propose three different transfer approaches in this paper, including *BestGen*, *Half-Transfer* and *Full-Transfer*. In Fig. 3, the training process of GPHH with three different approaches

on the source domain is presented. The training process on the target domain of the three transfer approaches is further summarized in Algorithm 1.



(a) BestGen: Transfer the best individuals from the Pareto Front of each generation

(b) Full-Transfer: Transfer the First and the Second Pareto Front

(c) Half-Transfer: Transfer the Pareto Front of the last generation with random trees

Fig. 3: The training process of three different approaches on source domain

BestGen BestGen in Algorithm 1 is a transfer learning approach that transfers some best individuals from the source domain to the target domain. It is different from the approach in [6] which was proposed for the single-objective optimisation problem. The transfer learning approach in [6] collects from 10% to 20% of the best individuals in each generation on the source domain, which cannot be used

Algorithm 1 Training process of GPHH with transfer learning

Input: a list of training instance on source domain S_1, a list of training instance on target domain S_2,
Output: a set of non-dominating heuristics on target domain

```
Randomly initialize the population P_1
while max generation not reached do
    Evolve offspring through selection, crossover and mutation
    Evaluate the fitness of all evolved offspring on source domain
end while
Identify the set of individuals I to transfer from source domain to target domain
```

$$I = \begin{cases} \text{Best of } k \text{ individuals in each generation,} & \textbf{BestGen} \\ 50\% \text{ from First Front} + 50\% \text{ from Second Front,} & \textbf{Full-Transfer} \\ 50\% \text{ from First Front rules,} & \textbf{Half-Transfer} \end{cases}$$

```
Initialize the population P_2 using a combination of I and randomly generated rules
while max generation not reached do
    Evolve offspring through selection, crossover and mutation
    Evaluate the fitness of all evolved offspring on target domain
end while
Return the Pareto Front of the last evolved population
```

to solve our multi-objective optimisation problem. In this paper we propose to select a pre-defined number of individuals from the Pareto Front of each generation. Since the number of individuals in the Pareto Front can be very large, the crowding distance [5] is applied to ensure our selected individuals keep most of the diversity of the Pareto Front. The selected individuals are then used to generate the initial population of the the target domain. The training process starts from a population with knowledge collected from the source domain.

Full-Transfer Full-Transfer approach in Algorithm 1 randomly selects half of the first Pareto Front and half of the second Pareto Front over one single run on source domain. The second Pareto Front refers to the evolved rules that are only dominated by the rules in the first Pareto Front and are not dominated by other rules in the second Pareto Front. In the training process on the source domain, we keep the individuals in the first Pareto Front and the second Pareto Front. The archive is updated by each generation. After the training process on the source domain, the initial population is generated from the archive that consists of the First and Second Pareto Front of the training process.

Half-Transfer Half-Transfer in Algorithm 1 is to transfer half of the final population from the source domain to the target domain. To solve our multi-objective problem we design an approach to selectively choose the solutions on the front from the final generation evolved in the source domain. The Half-Transfer approach selects the Pareto Front over one single run on source domain. The approach then randomly selects half of evolved brokering rules from the Pareto

Front obtained on the source domain to create the initial population for evolution on the target domain. The other half of the initial population is randomly generated to maintain the diversity.

Note that Full-Transfer approach uses the second Pareto Front to keep the diversity. Compared to the Half-Transfer approach, it transfers more knowledge from the source domain, since the second Pareto Front is significantly better than the randomly generated rules.

4.3 Crossover and Mutation

Three genetic operators, crossover, mutation, and selection, are applied in our transfer learning methods. In the crossover process, each parent individual is cut-off by a randomly selected point. Then the offspring can be generated by swapping the two parts starting from the cut-off points from their parents. To maintain the diversity, the mutation operation randomly generates a new branch at the mutation point of the parent individuals.

4.4 Fitness Evaluation

We use fitness values to evaluate each individual. Each pair of fitness values is computed based on eq. (3). Each individual can be represented as a mathematical function. Given the arrived requests, we can apply the function of each individual to sort the priority of candidate regions list and select the VM in the region that has the highest priority in the region list (see [4] for more details). After all requests are assigned, we calculate the sum of the cost TC and the average latency ANL as the fitness values for this individual.

5 Experiments

In this section, the experimental design is introduced. We perform the experiments to simulate the cloud environment in the real world. We also present our evaluation results on the generated rules using Hypervolume (HV) [27] and Inverted Global Distance (IGD) [14].

In our experiments, hypervolume (HV) describes the area that covers all area dominated by our proposed rules and the reference point (1,1) [10]. A higher value of HV represents a better diversity of our proposed solutions.

IGD describes the average distance from our proposed rules to the true Pareto Front. The true Pareto Front in this problem can be approximated by obtaining the Pareto Front from all non-dominated solutions over 30 independent runs. A lower value of IGD indicates a better diversity and convergence of our proposed solutions [17].

5.1 Simulation and Datasets

We use the simulator developed in [23] for multi-cloud service brokering to evaluate our proposed transfer learning approaches.

The simulator has the following features:

- For each VM type, the number of VM instances available is infinite.
- Requests arrive sequentially in a fixed timespan (one day).
- Once a request arrived, the request will be assigned immediately.
- All requests are equally important.
- Multiple VM types and multiple datacenters are available in the simulator.

The dataset that we used for training and testing are real world data, as used in [4]. 15 different VM types are included in our experiments, 5 from each service provider, Alibaba Elastic Compute Service (ECS), Amazon Web Services (AWS) and Microsoft Azure. As in [20, 21], we adopt 82 user locations in the Sprint IP Network[3] to simulate the global user community. To determine the network latency between users and assigned VM instances, we collect real-world observations of network latency from Sprint IP backbone network databases[4].

The source domain of the MOLSB problem with 8 different data centers, including Dublin, Singapore, Sydney, North Virginia, Mumbai, Tokyo, North California and Sao Paulo. Following [7], User requests arrive at the data centers with 1% arrival rate of the Microsoft Azure dataset. The target problem domain is the MOLSB problem with 15 different data centers, with 7 more regions including Frankfurt, Hong Kong, London, Paris, Seoul, Stockholm and Montreal.

To evaluate our proposed GPHH with transfer learning, we created 5 test cases of the target domain. Each test case contains one test set, which is unseen in training process. The performance of each test cases and the average performance of 5 test sets are calculated.

5.2 Baseline Algorithm

As mentioned in Section 1, there is no transfer learning method proposed for the MOLSB problem. To evaluate our proposed GPHH with three transfer learning approaches, namely BestGen, Half-Transfer, and Full-Transfer, we compare the transfer learning approaches with GPHH-MOLSB [4]. GPHH-MOLSB is trained on target domain directly without any knowledge transferred from the source domain.

5.3 Parameter Settings

In our experiments, we follow the GPHH parameters settings in the existing work [4] and [25]. In both the source and target domain, the population size is 1024 and the generation size is 100. The crossover rate and the mutation rate

[3] https://www.sprint.net
[4] https://www.sprint.net/tools/ip-network-performance

are 90% and 10%. The maximum depth is 7. In order to approximate the true Pareto Front, we train 30 independent runs with different random seeds. All transfer learning approaches are implemented using DEAP [11].

5.4 Results

Table 3 and Table 4 show the average HV and IGD results of GPHH with different transfer techniques and the baseline method without knowledge transferred, i.e., GPHH-MOLSB.

Table 3: Average HV result of 5 test cases

Test Case	GPHH-MOLSB [4]	BestGen	Half-Transfer	Full-Transfer
1	0.9800±0.0005	0.9809±0.0003	0.9810±0.0003	**0.9811±0.0002**
2	0.9798±0.0006	0.9802±0.0004	0.9803±0.0003	**0.9805±0.0003**
3	0.9796±0.0005	0.9801±0.0004	0.9803±0.0002	**0.9804±0.0002**
4	0.9779±0.0008	0.9787±0.0005	0.9788±0.0003	**0.9790±0.0002**
5	0.9786±0.0006	0.9794±0.0003	0.9795±0.0003	**0.9796±0.0002**
Average	0.9791±0.0006	0.9798±0.0004	0.9800±0.0003	**0.9801±0.0002**

As can be seen from Table 3, all three transfer techniques have better HV results than GPHH-MOLSB. For example, in test case 1, GPHH-MOLSB has a HV result of 0.9800. The HV results of BestGen, Half Transfer and Full Transfer approaches are 0.9809, 0.9810 and 0.9811. Similarly, in Table 4, the average IGD results of BestGen, Half Transfer and Full Transfer approaches are 0.00069, 0.00069 and 0.00065, while the average IGD of GPHH-MOLSB is 0.00112. In Table 3, the Full Transfer approach achieves the highest HV in all 5 test cases, while the BestGen approach has the lowest HV result.

Table 4: Average IGD result of 5 test cases

Test Case	GPHH-MOLSB [4]	BestGen	Half-Transfer	Full-Transfer
1	10.2223±2.4358e−4	6.2131±0.5317e−4	6.2117±0.4607e−4	**5.8971±0.5317**e−4
2	11.5345±3.0194e−4	6.7907±0.4728e−4	6.8737±0.4598e−4	**6.3855±0.5289**e−4
3	11.2997±2.8345e−4	6.8132±0.6295e−4	6.8763±0.5525e−4	**6.4603±0.5971**e−4
4	11.1337±3.2152e−4	7.0447±0.5916e−4	7.1584±0.5642e−4	**6.6599±0.4936**e−4
5	12.0743±3.3086e−4	7.5021±0.4838e−4	7.4144±0.4997e−4	**7.2684±0.5037**e−4
Average	11.2475±2.9627e−4	6.8727±0.5419e−4	6.9069±0.5074e−4	**6.5342±0.5310**e−4

In Table 4, the Full-Transfer approach has the lowest IGD in all 5 test cases, BestGen has the highest IGD in most of the test cases except test case

4. All three transfer learning approaches have achieved better performance than GPHH-MOLSB. This demonstrates that the three transfer learning methods are able to keep the diversity of the population, which is important for evolutionary processes.

The results in Table 3 and Table 4 show that the heuristics generated by three transfer learning approaches outperform the GPHH-MOLSB without transfer knowledge in terms of the IGD and HV results.

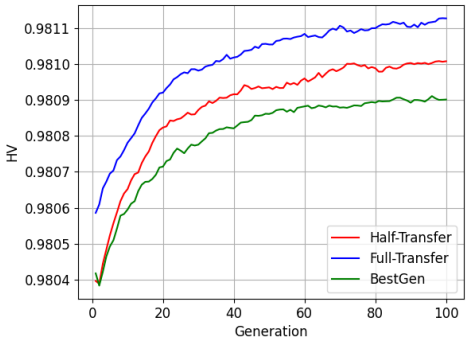


Fig. 4: Convergence curve of HV for test case 1 on 15 data centers

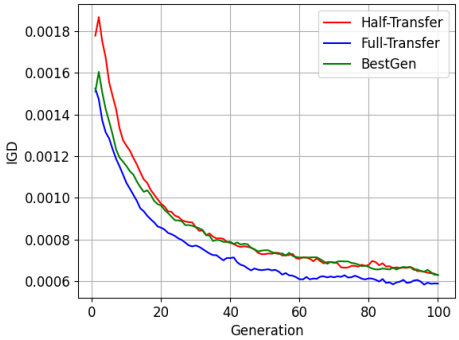


Fig. 5: Convergence curve of IGD for test case 1 on 15 data centers

To further investigate the effectiveness of the three transfer learning approaches during the training process of the source domain, we analyze the convergence curves of the three approaches. Fig. 4 and Fig. 5 present the convergence curves regarding the average HV and IGD obtained by three transfer learning

approaches on the test case 1 with 15 data centers. As seen in the two figures, all three approaches start with a high HV and a low IGD results. The HV results of all three transfer approaches increase rapidly, and the IGD results decrease significantly from the first generation to the 40-th generation. From the 80th generation to the last generation, the HV and IGD results change slightly with fluctuation in all three transfer approaches.

As seen in Fig.4 and Fig.5, among the three transfer learning methods, GPHH with Full-Transfer method performs the best through all the generations. The results demonstrate that knowledge obtained from the source domain can help GPHH to generate high quality rules in the target domain.

6 Conclusion

In this paper, we propose three multi-objective GPHH approaches with different transfer learning techniques, including Full-Transfer, Half-Transfer and Best-Gen, to solve the dynamic MOLSB problem. Our experimental evaluation using datasets collected from real world demonstrates that all the three approaches of GPHH with transfer learning outperform an existing approach without using transfer learning. All the three transfer learning approaches generate better heuristics with higher HV and lower IGD than the existing GPHH method.

References

1. Ardeh, M.A., Mei, Y., Zhang, M.: Genetic programming with knowledge transfer and guided search for uncertain capacitated arc routing problem. IEEE Transactions on Evolutionary Computation (2021)
2. Ardeh, M.A., Mei, Y., Zhangz, M.: Diversity-driven knowledge transfer for GPHH to solve uncertain capacitated arc routing problem. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 2407–2414. IEEE (2020)
3. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society **64**(12), 1695–1724 (2013)
4. Chen, Y., Shi, T., Ma, H., Chen, G.: Automatically design heuristics for multi-objective location-aware service brokering in multi-cloud. In: 2022 IEEE International Conference on Services Computing (SCC). pp. 206–214. IEEE (2022)
5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation **6**(2), 182–197 (2002)
6. Dinh, T.T.H., Chu, T.H., Nguyen, Q.U.: Transfer learning in genetic programming. In: 2015 IEEE Congress on Evolutionary Computation (CEC). pp. 1145–1151. IEEE (2015)
7. Du, B., Wu, C., Huang, Z.: Learning resource allocation and pricing for cloud profit maximization. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 7570–7577 (2019)
8. Durillo, J.J., Fard, H.M., Prodan, R.: Moheft: A multi-objective list-based method for workflow scheduling. In: 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings. pp. 185–192. IEEE (2012)

9. Escott, K.R., Ma, H., Chen, G.: Transfer learning assisted GPHH for dynamic multi-workflow scheduling in cloud computing. In: Australasian Joint Conference on Artificial Intelligence. pp. 440–451. Springer (2022)
10. Fonseca, C.M., Paquete, L., López-Ibánez, M.: An improved dimension-sweep algorithm for the hypervolume indicator. In: 2006 IEEE international conference on evolutionary computation. pp. 1157–1163. IEEE (2006)
11. Fortin, F.A., De Rainville, F.M., Gardner, M.A.G., Parizeau, M., Gagné, C.: Deap: Evolutionary algorithms made easy. The Journal of Machine Learning Research **13**(1), 2171–2175 (2012)
12. Heilig, L., Buyya, R., Voß, S.: Location-aware brokering for consumers in multi-cloud computing environments. Journal of Network and Computer Applications **95**, 79–93 (2017)
13. Iqbal, M., Xue, B., Al-Sahaf, H., Zhang, M.: Cross-domain reuse of extracted knowledge in genetic programming for image classification. IEEE Transactions on Evolutionary Computation **21**(4), 569–587 (2017)
14. Ishibuchi, H., Masuda, H., Tanigaki, Y., Nojima, Y.: Modified distance calculation in generational distance and inverted generational distance. In: International conference on evolutionary multi-criterion optimization. pp. 110–125. Springer (2015)
15. Koçer, B., Arslan, A.: Genetic transfer learning. Expert Systems with Applications **37**(10), 6997–7002 (2010)
16. Koza, J.R., Poli, R.: Genetic programming. In: Search methodologies, pp. 127–164. Springer (2005)
17. Ma, H., da Silva, A.S., Kuang, W.: NSGA-II with local search for multi-objective application deployment in multi-cloud. In: 2019 IEEE Congress on Evolutionary Computation (CEC). pp. 2800–2807. IEEE (2019)
18. Mansouri, Y., Toosi, A.N., Buyya, R.: Brokering algorithms for optimizing the availability and cost of cloud storage services. In: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science. vol. 1, pp. 581–589 (2013)
19. Shi, T., Ma, H., Chen, G.: A genetic-based approach to location-aware cloud service brokering in multi-cloud environment. In: 2019 IEEE International Conference on Services Computing (SCC). pp. 146–153. IEEE (2019)
20. Shi, T., Ma, H., Chen, G., Hartmann, S.: Location-aware and budget-constrained application replication and deployment in multi-cloud environment. In: 2020 IEEE International Conference on Web Services (ICWS). pp. 110–117. IEEE (2020)
21. Shi, T., Ma, H., Chen, G., Hartmann, S.: Location-aware and budget-constrained service deployment for composite applications in multi-cloud environment. IEEE Transactions on Parallel and Distributed Systems **31**(8), 1954–1969 (2020)
22. Shi, T., Ma, H., Chen, G., Hartmann, S.: Cost-effective web application replication and deployment in multi-cloud environment. IEEE Transactions on Parallel and Distributed Systems **33**(8), 1982–1995 (2021)
23. Shi, T., Ma, H., Chen, G., Hartmann, S.: Location-aware and budget-constrained service brokering in multi-cloud via deep reinforcement learning. In: International Conference on Service-Oriented Computing. pp. 756–764. Springer (2021)
24. Simarro, J.L.L., Moreno-Vozmediano, R., Montero, R.S., Llorente, I.M.: Dynamic placement of virtual machines for cost optimization in multi-cloud environments. In: International Conference on High Performance Computing Simulation. pp. 1–7 (2011)
25. Tan, B., Ma, H., Mei, Y.: A hybrid genetic programming hyper-heuristic approach for online two-level resource allocation in container-based clouds. In: 2019 IEEE Congress on Evolutionary Computation (CEC). pp. 2681–2688. IEEE (2019)

26. Weiss, K., Khoshgoftaar, T.M., Wang, D.: A survey of transfer learning. Journal of Big data **3**(1), 1–40 (2016)
27. While, L., Hingston, P., Barone, L., Huband, S.: A faster algorithm for calculating hypervolume. IEEE transactions on evolutionary computation **10**(1), 29–38 (2006)
28. Zhang, F., Mei, Y., Nguyen, S., Zhang, M.: Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. ieee transactions on cybernetics **51**(4), 1797–1811 (2020)
29. Zhang, F., Mei, Y., Nguyen, S., Zhang, M., Tan, K.C.: Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling. IEEE Transactions on Evolutionary Computation **25**(4), 651–665 (2021)