Particle Swarm Optimization for Multi-Objective Web Service Location Allocation

Boxiong Tan[1], Yi Mei[1], Hui Ma[1], Mengjie Zhang[1]

Victoria University of Wellington, Wellington, New Zealand
titantbx1989215@gmail.com
yi.mei@ecs.vuw.ac.nz
hui.ma@ecs.vuw.ac.nz
mengjie.zhang@ecs.vuw.ac.nz

Abstract. Web service location allocation problem is an important problem in the modern IT industry. In this paper, the two major objectives, i.e. deployment cost and network latency, are considered simultaneously. In order to solve this new multi-objective problem effectively, we adopted the framework of binary Particle Swarm Optimization (PSO) due to its efficacy that has been demonstrated in many optimization problems. Specifically, we developed two PSO variants, one with weighted-sum fitness function (WSPSO) and the other with dominance-based fitness function. Concretely, it uses the fast Non-dominate Sorting scheme, and thus is called NSPSO. The experimental results showed that both PSO variants performed better than NSGA-II, which is the one of the most commonly used multi-objective genetic algorithms. Furthermore, we have found that NSPSO achieved a more diverse set of solutions than WSPSO, and thus covers the Pareto front better. This demonstrates the efficacy of using the dominance-based fitness function in solving multi-objective Web service location allocation problem.

1 Introduction

The *Web Service Location Allocation Problem* (WSLAP) is a significant problem that is important for many modern IT enterprises. Given a set of *Web services* and *candidate locations*, WSLAP is to assign each Web service to at least one location (one or more copies) to optimize certain objective such as the total deployment cost and response time. To accommodate business agility, it is usually preferred to use existing applications instead of developing them from scratch. To this end, a contemporary approach is to package the software resources as Web services (e.g. in the service oriented architecture [6] [18]), which are well-defined, self-contained modules that provide standard business functionality and are independent of the state or context of other services [20]. It has been demonstrated that the Web service technology has the advantages of convenience, low cost and capacity to be composed into high-level business processes [1]. This

provides possibility of combining coarse-grained Web servies to build complex applications using standards such as WS-BPEL [17].

In practice, the Web services are generally located in some physical places (e.g. servers) by Web server providers, and can be called by users from various locations. In this situation, how to select proper locations for the Web services becomes an important problem. Therefore, one needs to assign the given Web services to proper locations. In WSLAP, in addition to the functionality requirement (e.g. the system can response to any type of requests), there are a number of *Quality of Service* (QoS) objectives for the Web service providers to consider to become competitive in the market. QoS, also known as non-functional requirements to Web services, is the degree to which a Web service meets specified requirements or user needs [25]. The common QoS measures include deployment cost, response time, security and availability. Web service location allocation has significant impact on two QoS measures, i.e. deployment cost and response time. Therefore, in this paper, we study Web service location allocation with two objectives, minimizing the deployment cost and network latency.

It is obvious that the two objectives are conflicting with each other. For example, to reduce the deployment cost, one needs to reduce the number of Web services deployed. This will increase the network latency due to the lack of services nearby. The deployment cost and network latency have been considered separately in literature [9] [12]. However, to the best of our knowledge, there is no study trying to minimize the cost and response time simultaneously.

In our study, we aim to solve the Multi-Objective WSLAP (MO-WSLAP) that minimizes the cost and response time simultaneously. Instead of providing a single solution, we expect to provide a set of trade-off solutions, which are so-called *Pareto optimal* solutions. Evolutionary algorithms are chosen to solve the problem since they maintain a population of individuals during the search, and thus are able to provide a set of solutions in a single run. To be more specific, the framework of binary Particle Swarm Optimization (PSO) was adopted here because it has been successfully applied to many real-world optimization problems.

In summary, our goals in the paper are given as follows.

1. The total deployment cost and network latency simultaneously are considered, which leads to a Multi-Objective WSLAP (MO-WSLAP);
2. Two binary PSO approaches are designed for solving the MO-WSLAP, considering different multi-objective fitness assignment schemes;
3. The efficacy of using binary PSO to solve the MO-WSLAP are verified by comparing with a well known multi-objective optimization algorithm (NSGA-II).

The rest of the paper is organized as follows: Section 2 introduces the background, including the problem description and related work. Then, the PSO algorithms proposed for solving MO-WSLAP is described in Section 3. The experimental studies are conducted in Section 4. Finally, the conclusions and future work are given in Section 5.

2 Background

2.1 Problem Description

In WSLAP, a set of *user centres* $\mathcal{U} = \{U_1, \ldots, U_m\}$ and a set of *candidate locations* $\mathcal{A} = \{A_1, \ldots, A_n\}$ are given. A user centre indicates a centre city of a user-concentrated area. A candidate location is the geographic location that is suitable to deploy the Web services, e.g. the locations of the existing Web server hosting providers. There is a pool of Web services $\mathcal{W} = \{W_1, \ldots, W_s\}$. Each Web service $W_i \in \mathcal{W}$ must be deployed to at least one location. Note that a Web service can have multiple copies that are located in different locations. For each Web service $W_i \in \mathcal{W}$ and each candicate location $A_j \in \mathcal{A}$, there is a deployment cost C_{ij} induced by deploying service W_i at location A_j. For each user centre $U_k \in \mathcal{U}$ and each candicate location $A_j \in \mathcal{A}$, there is a *latency* L_{jk}, which affects the *response time* from the location A_j to the user centre U_k. It mainly depends on the distance between the two geographical locations. For each Web service $W_i \in \mathcal{W}$ and each user centre $U_k \in \mathcal{U}$, there is an *invocation frequency* F_{ik}, indicating the frequency of the service W_i invoked by the users from U_k. Given all the above information, the problem is to design a plan to deploy the services, so that each service is deployed in one or more locations, and the *total deployment cost* f_1 and *network latency* f_2 of the system is minimized. The deployment cost and network latency can be calculated as follows:

$$f_1 = \sum_{i=1}^{s} \sum_{j=1}^{n} C_{ij} x_{ij}, \tag{1}$$

$$f_2 = \sum_{i=1}^{s} \sum_{k=1}^{m} F_{ik} r_{ik}, \tag{2}$$

where x_{ij} takes 1 if service W_i is assigned in location A_j, and 0 otherwise. r_{ik} stands for the response time of service W_i to the user centre U_k, which is calculated as

$$r_{ik} = \min\{L_{jk} \mid j \in \{1, \ldots, n\} \text{ and } x_{ij} = 1\}. \tag{3}$$

Then, the problem can be stated as follows:

$$\min\ f_1 = \sum_{i=1}^{s} \sum_{j=1}^{n} C_{ij} x_{ij}, \tag{4}$$

$$\min\ f_2 = \sum_{i=1}^{s} \sum_{k=1}^{m} F_{ik} r_{ik}, \tag{5}$$

$$s.t. : \sum_{j=1}^{n} x_{ij} \geq 1,\ \forall\ i \in \{1, \ldots, s\}, \tag{6}$$

$$x_{ij} \in \{0, 1\},\ \forall\ i \in \{1, \ldots, s\},\ \forall\ j \in \{1, \ldots, n\}. \tag{7}$$

Eqs. (4) an (5) are the two objectives to be minimized. Eq. (6) indicates that each service must be deployed in at least one location. Eq. (7) gives the domain of the decision variables x_{ij}.

2.2 Related Work

Most of the previous work treated WSLAP as a single objective problem. In [1] [22], integer linear programming techniques were used to solve the problem. In particular, the work in [22] solved the problem by employing greedy and linear relaxation.

Researches on network virtualization [2] [8] employed greedy algorithms to allocate virtual machines (VMs) in the data center so that the requirements of network bandwidth are met. [14] presented a multi-layer and integrated fashion through a convex integer programming formulation.

The major drawback of greedy algorithm is that it is easy to be stuck at local optima. On the other hand, it is well known that integer linear programming has a high complexity and thus does not scale well. It can only be used in small or medium sized problem instances. In this case, heuristics and meta-heuristics such as genetic algorithms are promising to achieve better solutions within a short time.

Huang [9] proposed an enhanced Genetic Algorithm (GA)-based approach for the problem. However, only network latency was considered in the paper. Kessaci [12] proposed a new multi-objective genetic algorithm called MOGA-CB for minimizing the cost of VMs instance and response time. A framework called Green Monster was proposed in [19] to dynamically move Web services across Internet data centres for reducing their carbon footprint while maintaining their performance. Green monster applied a modified version of NSGA-II [7] with an additional local search process.

In summary, although there have been a number of works trying to solve WSLAP in different ways, no work exists to consider minimizing the deployment cost and network latency simultaneously. Therefore, in this paper, we formulate the multi-objective model and attempt to solve it with PSO.

3 Particle Swarm Optimization for Multi-Objective Web Service Location Allocation

PSO was proposed by Kennedy and Eberhart in 1995 [10]. It is a simple yet powerful optimization algorithm that mimics the flock behavior to search in the solution space. It has been successfully applied to various optimzation problems. Thus, we adopt the PSO framework to solve the MO-WSLAP in this paper. The generic PSO framework is given in Algorithm 1.

In line 4, the personal best location of each particle is the location with the best objective value that the particle found so far. In line 5, the global best location is the best location among the personal best locations of all the particles.

Algorithm 1: The generic framework of PSO

```
1  Randomly generate an initial swarm and the velocities;
2  repeat
3      foreach particle i in the swarm do
4          Update the personal best location p_i;
5          Update the global best location g;
6      end
7      foreach particle i in the swarm do
8          Update particle velocity v_i;
9          Update and evaluate particle location x_i;
10     end
11 until termination criterion is met;
12 return the global best g;
```

In line 8, the standard way of updating each dimension v_{id} of the velocity $\boldsymbol{v}_i$ is as follows:

$$v_{id} \leftarrow w \cdot v_{id} + c_1 \cdot r_{1i} \cdot (p_{id} - x_{id}) + c_2 \cdot r_{2i} \cdot (g_d - x_{id}), \tag{8}$$

where w is the inertia weight, c_1 and c_2 are the acceleration factors, and r_{1i} and r_{2i} are random variables sampled from uniform distribution between 0 and 1. v_{id}, x_{id}, p_{id} and g_d stand for the value in dimension d of $\boldsymbol{v}_i$, $\boldsymbol{x}_i$, $\boldsymbol{p}_i$ and $\boldsymbol{g}$ respectively. In MO-WSLAP, the decision variables x_{ij} are binary variables, i.e. they can only take 0 or 1. In this case, we employ the Binary PSO (BPSO) [11]. In line 9, the location is updated as follows:

$$x_{id} = \begin{cases} 1, & \text{if } \mathit{rand}() < \frac{1}{1+e^{-v_{id}}}, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

where $\mathit{rand}()$ is a value sampled from the uniform distribution between 0 and 1.

When solving MO-WSLAP, problem-specific solution representation, fitness evaluation and constraint handling must be designed. They will be described one by one in this section.

3.1 Solution Representation

In MO-WSLAP, the decision variables are x_{ij}, where $i = 1, \ldots, s$, and $j = 1, \ldots, n$. That is, the decision variables form a $s \times n$ matrix. Note that PSO was designed for vector-based solutions. Therefore, we simply adopt the flatten matrix representation that transforms the matrix $\boldsymbol{X}_{s \times n}$ into a $(s \times n)$–dimensional vector $\boldsymbol{y}$. The element x_{ij} in $\boldsymbol{X}$ corresponds to the $(n \cdot (i-1) + j)^{th}$ element in $\boldsymbol{y}$. For example, given a 3×3 matrix as follows:

$$\boldsymbol{X} = \begin{bmatrix} 0\ 1\ 0 \\ 0\ 0\ 1 \\ 1\ 0\ 0 \end{bmatrix},$$

the flatten matrix (vector) is $\boldsymbol{y} = (0, 1, 0, 0, 0, 1, 1, 0, 0)$. A $(s \times n)$–dimensional velocity vector $\boldsymbol{v}$ is defined accordingly, each for an element $y \in \boldsymbol{y}$. The vector $\boldsymbol{y}$ is used in the update phase. Then, during the fitness evaluation, $\boldsymbol{y}$ is first decoded into the original matrix $\boldsymbol{X}$.

3.2 Fitness Evaluation

After decoding $\boldsymbol{y}$ into the corresponding matrix $\boldsymbol{X}$, the total deployment cost and network latency can be directly calculated by Eqs. (1) and (2). In MO-WSLAP, since the two objectives are considered simultaneously, it is important to normalize them so that they have the same scale. To this end, the lower and upper bounds of both the total deployment cost and network latency are calculated. Specifically, for the total deployment cost, the lower bound $f_{1,\min}$ is obtained by deploying each Web service once in the location that leads to the minimal deployment cost, while the upper bound $f_{1,\max}$ is obtained by employing each Web service in all the locations. For the network latency, the lower bound $f_{2,\min}$ is achieved by deploying all the Web services in all the locations, and the upper bound $f_{2,\max}$ is obtained by an exhaustive search in which each Web service is allocated in only one location. The search space to find $f_{2,\max}$ is $s \times n$.

Based on the bounds of f_1 and f_2, the linear normalization is conducted to obtain the normalized objective values, which are denoted as $\hat{f}_1$ and $\hat{f}_2$.

There are various ways for fitness assignment in evolutionary multi-objective optimization [3] [5] [7] [13] [15] [16]. In this paper, two different fitness assignment schemes are selected and compared. The first one is the weighted sum aggregating function, which is the most straightforward way that combines multiple objectives into a single one. The resultant PSO is called the Weighted Sum PSO (WSPSO). In WSPSO, the fitness function is defined as follows:

$$fitness = \lambda \cdot \hat{f}_1 + (1 - \lambda) \cdot \hat{f}_2. \tag{10}$$

That is, particle $\boldsymbol{y}_1$ is considered to be better than particle $\boldsymbol{y}_2$, if $fitness(\boldsymbol{y}_1) < fitness(\boldsymbol{y}_2)$. Note that the fitness value depends not only on $\hat{f}_1$ and $\hat{f}_2$, but also the weight coefficient λ. In the experimental studies, λ is simply set to 0.5 as a rule of thumb. That is, the two objectives are of the same importance. However, in practice this value can be obtained from service providers.

The second strategy is the dominance-based fitness assignment scheme, and the resultant PSO is called the Non-Dominated PSO (NSPSO) [15]. Given a set of objective functions, solution x_1 is said to *dominate* solution x_2, if (1) x_1 is no worse than x_2 in all the objectives, and (2) x_1 is better than x_2 in at least one objective. Based on the dominance relation, Deb et al. [7] designed a fast non-dominated sorting procedure, which is efficient in sorting a population of individuals from the best to the worst. The basic idea is to first divide the individuals into different fronts. The first front consists of all the individuals that are not dominated by any other individuals in the population. Then, each subsequent front includes the individuals that are dominated by no other individuals than those in the previous fronts. Within the same front, a *crowding*

distance measure is designed to measure the crowdedness around each individual. Then, the individual with a larger crowding distance is considered to be in a less crowded region, and thus be better. Details of the fast non-dominated sorting can be found in [7].

NSPSO adopts the fast non-dominated sorting in fitness assignment. The differences between NSPSO and the standard PSO framework are mainly twofold. First, in each generation, instead of replacing the original particle, the update of each particle creates a new particle. After all the particles have been updated, all the original particles and newly created particles are combined together and sorted by the fast non-dominated sorting. Then, a new swarm is formed by selecting the first particles in the sorted set. Second, instead of choosing the location with the best fitness value, the global best location is chosen to be the one in the first front and with the least crowding distance value. If there are multiple such locations, one is randomly selected.

3.3 Constraint Handling

In MO-WSLAP, each service must be allocated to at least one location. However, during the search process, the constraint can be violated, and there might exist some services that are not allocated to any location. That is, infeasible particle may occur. There are a variety of strategies to handle the constraints [4]. Here, we employ the simplest constraint handling approach, which is to ignore all the infeasible particles. To this end, all the infeasible particles are assigned the highest possible objective values (1 after normalization). Specifically,

$$\hat{f}_u(\boldsymbol{y}) = \begin{cases} \hat{f}_u(\boldsymbol{y}), & \text{if } \boldsymbol{y} \text{ is feasible,} \\ 1, & \text{otherwise.} \end{cases}, \ u = 1, 2. \tag{11}$$

where $\hat{f}_u(\boldsymbol{y})$ stands for the u^{th} normalized objective value.

4 Experimental Studies

To verify the efficacy of the proposed WSPSO and NSPSO, we tested them on the real-world network datasets, and compared with NSGA-II [7].

4.1 Datasets

The network dataset provided by WS-DREAM [23] [24] is used in the experiments. The dataset includes 339 user centres and 5825 candidate locations, along with the latency matrix between the user centres and the candidate locations. From Section 2.1, it is known that apart from the latency matrix, a MO-WSLAP instance consists of the other two matrices, i.e. the deployment cost matrix $\boldsymbol{C} = (C_{ij})_{s\times n}$ and invocation frequency matrix $\boldsymbol{F} = (F_{ik})_{s\times m}$. The matrices were generated as follows.

Deployment Cost: The deployment cost can include the fixed deployment fees (e.g. monthly rent) and variable fees (e.g. extra charges for exceeded storage and other limits) [21]. For the sake of simplicity, here we only considered the fixed deployment fees, which is independent of the location of the service. For each service, the deployment cost was randomly generated from a normal distribution with mean of 100 and standard deviation of 20.

Invocation Frequency: For each user centre and each service, the invocation frequency was randomly generated from a uniform distribution between 1 and 120.

In the experiments, we randomly generated a number of services, and selected different subsets of the latency matrix to form a set of different problem instances. The features of the generated instances are given in Table 1.

Table 1: The features of the generated problem instances.

Instance	**#Services**	**#Locations**	**#Centres**
Instance 1	20	5	10
Instance 2	20	10	10
Instance 3	50	15	20
Instance 4	50	15	40
Instance 5	50	25	20
Instance 6	50	25	40
Instance 7	100	15	20
Instance 8	100	15	40
Instance 9	100	25	20
Instance 10	100	25	40
Instance 11	200	25	40
Instance 12	200	25	80
Instance 13	200	40	40
Instance 14	200	40	80

4.2 Experiment Settings

In both WSPSO and NSPSO, the population size was set to 50, and the maximal number of generations was set to 50. c_1 and c_2 were set equally to 2, and w was set to 1. In addition, NSGA-II was taken into account for comparison. NSGA-II uses the tournament selection to select parents, and single point crossover and flip mutation operators to generate offsprings. In NSGA-II, the population size and maximal number of generations was set the same as that of the PSO approaches to make a fair comparison. The crossover and mutation rates were set to 0.8 and 0.2 respectively. For each instance and each algorithm, 40 independent runs were conducted.

4.3 Performance Measures

The Hypervolume and Inverted Generational Distance (IGD) indicators were chosen as the performance measures. The hypervolume is the area of the region in the objective space dominated by the given set of solutions. A larger hypervolume value indicates a better solution set. The IGD value is defined as the average distance from a Pareto-optimal solution to the closest solution in the given solution set. A smaller IGD value indicates that the given solution set is closer to the true Pareto front, and thus is better. The optimal IGD value is 0. The performance measures were calculated based on the normalized objective values. For the hypervolume, the nadir point was set to $(1, 1)$. For the IGD, since the true Pareto front is unknown, we selected the non-dominated solutions among the final solutions obtained by all the runs of WSPSO, NSPSO and NSGA-II as the approximation of the true Pareto front.

4.4 Results and Discussions

Performance of WSPSO: First, we analyse the performance of WSPSO, since it can be seen as the basic PSO version for solving MO-SWLAP. Table 2 shows the mean and standard deviation of the hypervolume of WSPSO on the tested instances. From the table, one can see that as the problem size increases, the performance of WSPSO decreased (the hypervolume value dropped).

Table 2: The mean and standard deviation of the hypervolume of WSPSO on the tested instances.

Instance 1	**Instance 2**	**Instance 2**	**Instance 4**	**Instance 5**
0.89 ± 0.016	0.61 ± 0.01	0.69 ± 0.007	0.71 ± 0.008	0.67 ± 0.007
Instance 6	**Instance 7**	**Instance 8**	**Instance 9**	**Instance 10**
0.63 ± 0.005	0.63 ± 0.006	0.65 ± 0.008	0.62 ± 0.005	0.58 ± 0.005
Instance 11	**Instance 12**	**Instance 13**	**Instance 14**	
0.55 ± 0.003	0.56 ± 0.003	0.56 ± 0.004	0.57 ± 0.003	

Fig. 1 shows the convergence curve of WSPSO on Instance 4. All the other instances showed a similar pattern. From the figure, it is clear that WSPSO converged well during the search process. More importantly, it seems that 50 generations is not sufficient for WSPSO to converge since the convergence curve is still dropping at generation 50. Note that MO-WSLAP is essentially a multi-objective optimization problem, and thus the weight sum fitness value is not comprehensive. Therefore, it is necessary to observe the temporal behavior of WSPSO in terms of both objectives rather than the aggregated fitness.

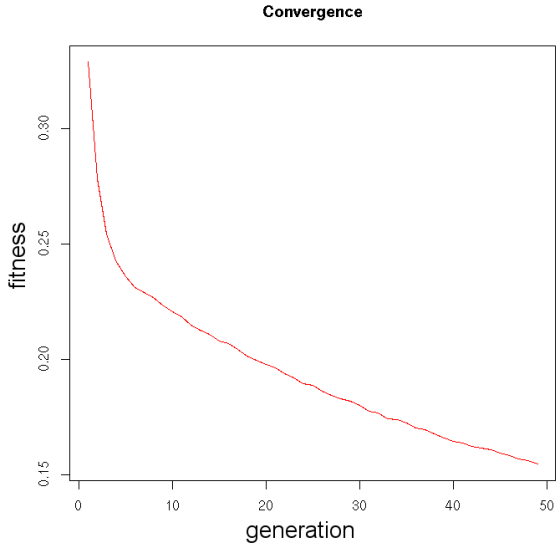


Fig. 1: The convergence curve (average fitness of the global best over 40 independent runs) of WSPSO on Instance 4.

Fig. 2 shows the evolution trajectory of all the particles in the 2D objective space at different stages of the search process on Instance 4. From the figure, one can see that the particles in generation 50 do not dominate those in generation 10. Instead, they have a different trade-off between the total deployment cost and network latency. More specifically, from generation 10 to generation 50, the total deployment cost is reduced, while the network latency increases. It can be seen that as early as in generation 10, the network latency already achieved the best value (0.025 after normalization). In contrast, the total deployment cost is much harder to improve (around 0.5 after normalization). Thus, much more effort has been put in improving the total deployment cost with a slight sacrifice in the network latency. Therefore, the downward convergence curve in Fig. 1 does not indicate that the particles are improved in both objectives. Instead, the total deployment cost is improved and the network latency is deteriorated, but to less extent.

Finally, although WSPSO is a single-objective optimization algorithm, there are a swarm of particles maintained in the last generation, from which one may still obtain the non-dominated set. To observe the distribution of the particles in the final swarm, we draw the scatter plot of the particles in the last generation of one run in the objective space on Instance 4, as shown in Fig. 3. All the other instances showed a similar pattern. It can be seen that there is still a non-dominate set in the final swarm, and the best particle with the minimal fitness value is the one with the minimal total deployment cost. This partly shows the capability of WSPSO in obtaining a set of trade-off solutions for MO-WSLAP.

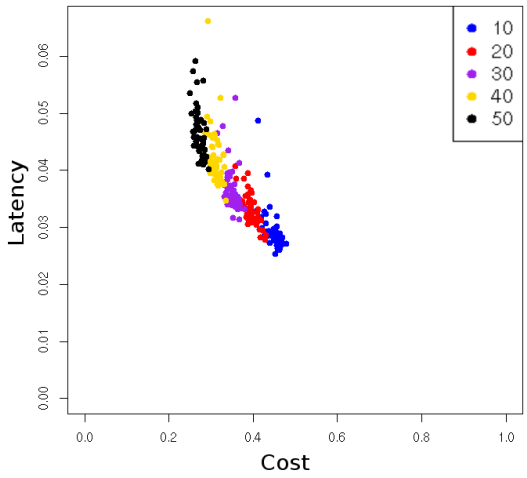


Fig. 2: The objective values of the particles at generation 10, 20, 30, 40 and 50 of WSPSO on Instance 4.

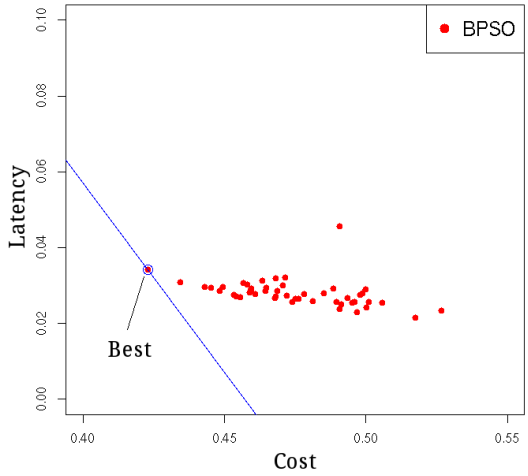


Fig. 3: The distribution of the final swarm in the objective space for Instance 4.

Performance of NSPSO: Then we evaluate the performance of NSPSO by comparing with WSPSO and NSGA-II. Tables 3 and 4 shows the mean and standard deviation of the hypervolume and IGD values of WSPSO, NSGA-II and NSPSO on the tested instances. The Wilxocon's rank sum test was conducted between each pair of the three compared algorithms with significance level of 0.05. For each instance, if one algorithm is significantly better than the other two, then the corresponding entry is marked in bold.

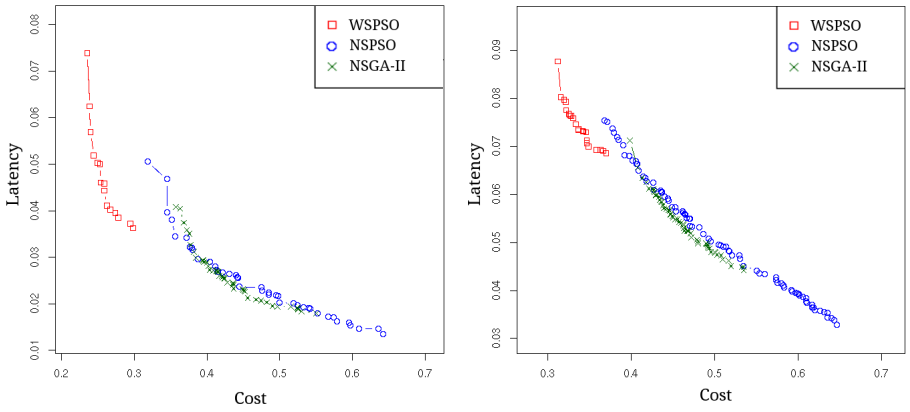
Table 3: The mean and standard deviation of the hypervolume of WSPSO, NSGA-II and NSPSO on the tested instances. If an algorithm significantly outperform the other two algorithms with significance level of 0.05, then the corresponding entry is marked in bold.

Instance	**WSPSO**	**NSGA-II**	**NSPSO**
Instance 1	**0.895 ± 0.0132**	0.828 ± 0.0129	0.757 ± 0.0187
Instance 2	**0.615 ± 0.0105**	0.606 ± 0.0123	0.596 ± 0.0108
Instance 3	**0.690 ± 0.0072**	0.594 ± 0.0074	0.615 ± 0.0111
Instance 4	**0.713 ± 0.0077**	0.606 ± 0.0074	0.627 ± 0.0112
Instance 5	**0.668 ± 0.0077**	0.575 ± 0.0047	0.605 ± 0.0093
Instance 6	**0.626 ± 0.0049**	0.554 ± 0.0063	0.587 ± 0.0070
Instance 7	**0.633 ± 0.0060**	0.564 ± 0.0048	0.600 ± 0.0083
Instance 8	**0.652 ± 0.0070**	0.577 ± 0.0051	0.614 ± 0.0079
Instance 9	**0.620 ± 0.0052**	0.555 ± 0.0039	0.597 ± 0.0089
Instance 10	**0.585 ± 0.0049**	0.535 ± 0.0049	0.579 ± 0.0079
Instance 11	0.554 ± 0.0030	0.520 ± 0.0026	**0.571 ± 0.0087**
Instance 12	0.562 ± 0.0031	0.523 ± 0.0033	**0.578 ± 0.0094**
Instance 13	0.563 ± 0.0028	0.529 ± 0.0029	**0.585 ± 0.0077**
Instance 14	0.566 ± 0.0031	0.533 ± 0.0026	**0.588 ± 0.0089**

Table 4: The mean and standard deviation of the IGD of WSPSO, NSGA-II and NSPSO on the tested instances. If an algorithm significantly outperform the other two algorithms with significance level of 0.05, then the corresponding entry is marked in bold.

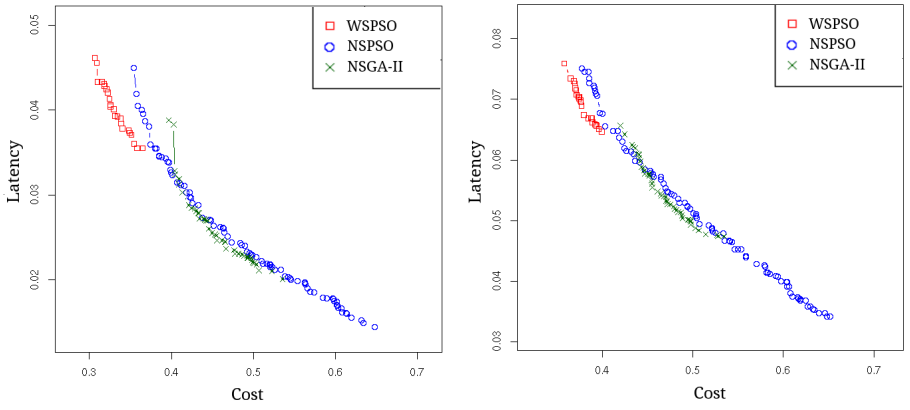
Instance	**WSPSO**	**NSGA-II**	**NSPSO**
Instance 1	0.194 ± 0.0176	**0.066 ± 0.0114**	0.092 ± 0.0110
Instance 2	0.146 ± 0.0131	**0.049 ± 0.0047**	0.057 ± 0.0035
Instance 3	0.146 ± 0.0089	0.036 ± 0.0029	**0.023 ± 0.0028**
Instance 4	0.133 ± 0.0089	0.038 ± 0.0028	**0.026 ± 0.0026**
Instance 5	0.110 ± 0.0060	0.037 ± 0.0030	**0.017 ± 0.0022**
Instance 6	0.123 ± 0.0068	0.036 ± 0.0040	**0.014 ± 0.0014**
Instance 7	0.128 ± 0.0091	0.037 ± 0.0036	**0.011 ± 0.0015**
Instance 8	0.114 ± 0.0068	0.041 ± 0.0036	**0.013 ± 0.0018**
Instance 9	0.100 ± 0.0045	0.039 ± 0.0025	**0.007 ± 0.0016**
Instance 10	0.104 ± 0.0046	0.041 ± 0.0037	**0.008 ± 0.0013**
Instance 11	0.077 ± 0.0038	0.036 ± 0.0018	**0.005 ± 0.0007**
Instance 12	0.074 ± 0.0033	0.035 ± 0.0021	**0.004 ± 0.0005**
Instance 13	0.076 ± 0.0027	0.044 ± 0.0024	**0.004 ± 0.0005**
Instance 14	0.069 ± 0.0023	0.039 ± 0.0022	**0.004 ± 0.0005**

From the tables, one can see that in terms of hypervolume, WSPSO performed significantly the best on the first 10 instances. NSPSO performed significantly better on the large scale instances (from 11 to 14). In terms of IGD, it is clear that NSPSO obtained significantly better values than WSPSO and NSGA-



(a) Instance 4 (b) Instance 6

Fig. 4: The best results of WSPSO, NSGA-II and NSPSO.

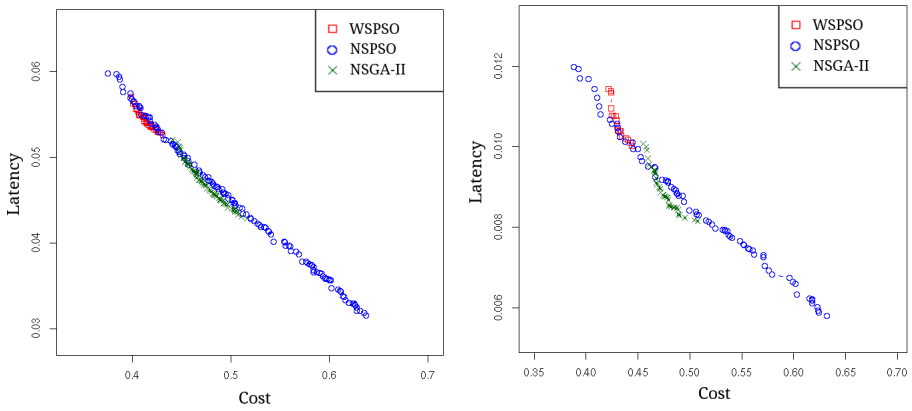


(a) Instance 8 (b) Instance 10

Fig. 5: The best results of WSPSO, NSGA-II and NSPSO.

II on 12 out of the total 14 instances (except the first two smallest instances, where NSGA-II performed better). Overall, NSPSO performed much better than the other two compared algorithms in terms of IGD. To better understand the inconsistency between the relative performance in terms of hypervolume and IGD, we plot the best results (the non-dominated solutions among those obtained in all the runs) of the compared algorithms in the objective space.

Figs. 4–6 show the results on Instances 4, 6, 8, 10, 12 and 14. From the figures, one can see that in general, NSPSO covered a much wider region than WSPSO and NSGA-II, especially when the problem size becomes larger. For the small sized Instance 4, the advantage of NSPSO is not obvious, as it cannot



(a) Instance 12 (b) Instance 14

Fig. 6: The best results of WSPSO, NSGA-II and NSPSO.

reach the region of WSPSO, which has a smaller deployment cost. Nevertheless, NSPSO still obtained the smallest IGD value due to the better coverage. As the problem size increases, the advantage of NSPSO becomes more and more obvious. As shown in Fig. 5, on Instances 8 and 10, NSPSO nearly covered the entire area obtained by NSGA-II and the area that is slightly worse than that of WSPSO where the deployment cost is between 0.3 and 0.4. More importantly, NSPSO reached the area where the network latency is lower than 0.02 and 0.045 respectively, which was never reached by WSPSO or NSGA-II. As a result, the solutions obtained by NSPSO played a major role in the approximated Pareto front. Since the IGD value highly depends on the approximated Pareto front, it is not surprising that NSPSO obtained significantly better IGD value. The solutions obtained by WSPSO and NSGA-II covered different areas in the objective space, which are non-overlapping with each other. The results of NSGA-II had a better spread, and thus contributed more in the approximated Pareto front. Thus, NSGA-II obtained better IGD values than WSPSO. Fig. 6 shows a similar pattern as Fig. 5. In this figure, the advantage of NSPSO is more obvious, as it nearly covered the areas of both NSGA-II and WSPSO. Therefore, it achieved both significantly better hypervolume and IGD values.

In summary, we have the following major observations.

- Both the proposed WSPSO and NSPSO performed well in solving MO-WSLAP. WSPSO performed better in terms of hypervolume, and NSPSO performed better in terms of IGD;
- Both the proposed WSPSO and NSPSO performed better than NSGA-II, which is the most commonly used multi-objective optimization algorithms;
- In MO-WSLAP, the relative performances of the compared algorithms are inconsistent in hypervolume and IGD. This is mainly due to the different optimization difficulties of the two objectives. Specifically, the total deploy-

ment cost is much harder to optimize than the network latency. As a result, the final solutions still have different scales in the two objectives after normalization. For example, as shown in Fig. 6b, for the final solutions, the deployment cost ranges from 0.4 to 0.6 and the network latency ranges from 0.006 to 0.012 on Instance 14. This makes the hypervolume emphasize more on the improvement in the deployment cost rather than the network latency.

5 Conclusions and Future Work

In this paper, the Multi-Objective Web Service Location Allocation Problem (MO-WSLAP) is investigated. In MO-WSLAP, the total deployment cost and network latency are to be minimized simultaneously. Two PSO variants are proposed for solving MO-WSLAP. One uses the weighted sum fitness evaluation, and the other uses the dominance-based fitness assignment. Both the proposed PSO methods performed better than the compared NSGA-II. Moreover, NSPSO performed better than WSPSO especially on larger instances. This demonstrates the efficacy of using dominance-based fitness assignment in solving real-world instances.

In the future, we plan to further improve the search scheme of PSO by employing more domain knowledge of the problem (e.g. the different optimization difficulties of the two objectives), and develop new PSO algorithms that can overcome the weaknesses of both WSPSO and NSPSO. We expect that the new PSO approach will be able to cover the areas of both WSPSO and NSPSO, and may reach a wider area in the objective space.

References

1. Aboolian, R., Sun, Y., Koehler, G.J.: A locationallocation problem for a web services provider in a competitive market. European Journal of Operational Research 194(1), 64 – 77 (2009)
2. Ballani, H., Costa, P., Karagiannis, T., Rowstron, A.: Towards predictable datacenter networks. ACM SIGCOMM (2011)
3. Coello, C., Pulido, G., Lechuga, M.: Handling multiple objectives with particle swarm optimization. Evolutionary Computation, IEEE Transactions on 8(3), 256–279 (2004)
4. Coello, C.A.C.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer methods in applied mechanics and engineering 191(11), 1245–1287 (2002)
5. Coello, C.A.C.: Evolutionary multi-objective optimization: a historical view of the field. Computational Intelligence Magazine, IEEE 1(1), 28–36 (2006)
6. Dan, A., Johnson, R.D., Carrato, T.: Soa service reuse by design. In: Proceedings of the 2Nd International Workshop on Systems Development in SOA Environments. pp. 25–28. SDSOA '08, ACM (2008)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. Evolutionary Computation, IEEE Transactions on 6(2), 182–197 (2002)

8. Guo, C., Lu, G., Wang, H., Yang, S., Kong, C., Sun, P., Wu, W., Zhang, Y.: Secondnet: A data center network virtualization architecture with bandwidth guarantees. In: ACM CONEXT 2010. Association for Computing Machinery, Inc. (2010)
9. Huang, H., Ma, H., Zhang, M.: An enhanced genetic algorithm for web service location-allocation. In: Decker, H., Lhotsk, L., Link, S., Spies, M., Wagner, R. (eds.) Database and Expert Systems Applications, Lecture Notes in Computer Science, vol. 8645, pp. 223–230. Springer International Publishing (2014)
10. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Neural Networks, 1995. Proceedings., IEEE International Conference on. vol. 4, pp. 1942–1948 vol.4 (1995)
11. Kennedy, J., Eberhart, R.: A discrete binary version of the particle swarm algorithm. In: Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on. vol. 5, pp. 4104–4108 vol.5 (1997)
12. Kessaci, Y., Melab, N., Talbi, E.G.: A pareto-based genetic algorithm for optimized assignment of vm requests on a cloud brokering environment. In: Evolutionary Computation (CEC), 2013 IEEE Congress on. pp. 2496–2503 (2013)
13. Knowles, J.D., Corne, D.W.: Approximating the nondominated front using the pareto archived evolution strategy. Evolutionary computation 8(2), 149–172 (2000)
14. Larumbe, F., Sanso, B.: Optimal location of data centers and software components in cloud computing network design. In: Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on. pp. 841–844 (2012)
15. Li, X.: A non-dominated sorting particle swarm optimizer for multiobjective optimization. In: Genetic and Evolutionary Computation – GECCO 2003, pp. 37–48 (2003)
16. Mei, Y., Tang, K., Yao, X.: Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem. IEEE Transactions on Evolutionary Computation 15(2), 151–165 (2011)
17. Organization for the Advancement of Structured Information Standards (OASIS): Web Services Business Process Execution Language (WS-BPEL) Version 2.0 (2007)
18. Papazoglou, M.P., Heuvel, W.J.: Service oriented architectures: Approaches, technologies and research issues. The VLDB Journal 16(3), 389–415 (2007)
19. Phan, D.H., Suzuki, J., Carroll, R., Balasubramaniam, S., Donnelly, W., Botvich, D.: Evolutionary multiobjective optimization for green clouds. In: Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation. pp. 19–26. GECCO '12, ACM (2012)
20. Ran, S.: A model for web services discovery with QoS. SIGecom Exch. 4(1), 1–10 (2003)
21. Sun, Y.: A Location Model for Web Services Intermediaries. Ph.D. thesis (2003), aAI3120151
22. Sun, Y., Koehler, G.J.: A location model for a web service intermediary. Decis. Support Syst. 42(1), 221–236 (2006)
23. Zhang, Y., Zheng, Z., Lyu, M.: Exploring latent features for memory-based QoS prediction in cloud computing. In: Reliable Distributed Systems (SRDS), 2011 30th IEEE Symposium on. pp. 1–10 (2011)
24. Zheng, Z., Zhang, Y., Lyu, M.: Distributed QoS evaluation for real-world web services. In: Web Services (ICWS), 2010 IEEE International Conference on. pp. 83–90 (2010)
25. Zhou, J., Niemela, E.: Toward semantic QoS aware web services: Issues, related studies and experience. In: Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on. pp. 553–557 (2006)