Evolutionary Multi-Objective Optimization for Web Service Location Allocation Problem

Boxiong Tan, Hui Ma, Yi Mei, and Mengjie Zhang

Abstract—With the ever increasing number of functionally similar web services being available on the Internet, the market competition is becoming intense. Web service providers (WSPs) realize that good Quality of Service (QoS) is a key of business success and low network latency is a critical measurement of good QoS. Because network latency is related to location, a straightforward way to reduce network latency is to allocate services to proper locations. However, Web Service Location Allocation Problem (WSLAP) is a challenging task since there are multiple objectives potentially conflicting with each other and the solution search space has a combinatorial nature. In this paper, we consider minimizing the network latency and total cost simultaneously and model the WSLAP as a multi-objective optimization problem. We develop a new PSO-based algorithm to provide a set of trade-off solutions. The results show that the new algorithm can provide a more diverse range of solutions than the compared three well known multi-objective optimization algorithms. Moreover, the new algorithm performs better especially on large problems.

Index Terms—Web Service Location Allocation, Quality of Service, Evolutionary Computation, Particle Swarm Optimization.

1 INTRODUCTION

Recently, companies have been employing service-oriented computing (SOC) developed softwares in an agile and cost efficient way [1]. As the building blocks of SOC, web services are well-defined, self-contained modules that provide standard business functionality and customers can access them via the Internet [2]. Web service providers always face a tradeoff while trying to improve the Quality of Service (QoS) [3] (e.g. service response time) and reduce the total cost. This study proposes a new algorithm to solve cost–QoS trade-off. Specifically, we use response time as the QoS criteria. Service providers can choose the most suitable allocation according to their budget and required service response time.

Web service allocation affects both service response time and total cost. Deploying services to user concentrated locations improves service response time largely because it reduces network latency [4]. On the other hand, deploying services at multiple locations invariably raised the total cost. Hence, the Web Service Location Allocation Problem (WSLAP) is essentially a multi-objective optimization problem [5] with two conflicting objectives, minimizing the response time and minimizing the total cost. Therefore, solutions of WSLAP have no single global optimum but a set of Pareto-optimal solutions. Although much research [6] discusses location-awareness, WSLAP has not been well studied.

WSLAP is an extension of P median problem [7] that has been proved as an NP-hard problem. WSLAP has a combinatorial search space. For example, to deploy 8 services at 4 locations. Each service can be deployed at multiple locations. Then, the number of deployment solution is $2^8 \times 4$. It is difficult for exhaustive algorithms to find optimal solutions in such a huge search space.

Current studies have two drawbacks. First, Aboolian and Sun [8], [9] treat the WSLAP as a single-objective problem and use traditional approaches, e.g. integer linear programming (ILP), to solve the problem. The algorithm generates only one solution based on the assumption that we know service providers’ preferences on budget and required response time. Second, their methods do not scale [10].

Multi-objective evolutionary algorithms (MOEAs) can address the above drawbacks effectively. First, MOEAs maintain a population of solutions during the search, and thus are able to obtain a set of non-dominated solutions in a single run. Second, MOEAs scale much better than ILPs due to the nature of heuristic search.

To address WSLAP as a multi-objective problem, we first proposed an aggregation approach with Binary Particle Swarm Optimization (BPSO) [11]. Even though the BPSO approach can solve the problem, a single-objective algorithm can only provide one solution for each run, hence, BPSO cannot provide alternatives when service providers do not have preferences. Therefore, we further investigated a Pareto front approach [12] with Sorting Genetic Algorithm-II (NSGA-II).

Solutions from BPSO and NSGA-II suggest that the proposed multi-objective approaches suit the problem well. This is because BPSO and NSGA-II are population-based algorithms. MOEA methodologies are ideal [13] for solving combinatorial optimization problems [14], [15].

However, we also found a disadvantage in both BPSO and NSGA-II. Their performances (in terms of convergence) drop rapidly when the problem size increases. In addition, the diversity of solutions is not maintained when problem size grows.

Hence this study focuses on how to maintain the performance when problem size increases. We consider a Multi-Objective Particle Swarm Optimization with Crowding Di-
tance (MOPSOCD). Raquel et al [16] propose MOPSOCD which can produce a well-distributed set of non-dominated solutions. MOPSOCD has two desired features: an external archive set and a mutation operator. These two features enhance the ability to avoid being stuck at local optima and maintaining a distributed non-dominated set.

However, the major obstacle is that MOPSOCD is a continuous optimization algorithm whereas WSLAP is a binary problem. Therefore, we develop a new binary version - BMOPSOCD. To this end, we use rounding functions to transform continuous values into binary ones. We propose three types of rounding functions and study their effects.

The overall goal is to develop a new binary multi-objective PSO-based approach to the WSLAP by considering two potentially conflicting objectives - minimizing cost and objective functions. It can be stated as follows:

2.2 Evolutionary Multi-objective Optimization

The Evolutionary Computation (EC) method is good at solving multi-objective optimization problems due to its capability of maintaining a set of solutions in its population. It finds a set of Pareto-optimal solutions in a single run. Evolutionary Multi-objective Optimization (EMO) has been extensively studied, and numerous EMO algorithms have been proposed.

NSGA-II [19] is a widely used multi-objective algorithm. NSGA-II proposes two innovative approaches: fast non-dominated sorting and crowding distance comparison-based diversity preservation. Strength Pareto Evolutionary Algorithm (SPEA2) [20] uses an external archive to store non-dominated solutions and a nearest neighbor density estimation technique to guide the search process. MOEA/D [21] decomposes a multi-objective problem into a number of scalar optimization subproblems and optimizes them simultaneously.

2.3 Particle Swarm Optimization (PSO)

Kennedy and Eberhart proposed PSO in 1995 [22]. It is a meta-heuristic algorithm inspired by the social behavior of birds. In PSO, each individual — a particle — searches the solution space. The underlying phenomenon of PSO is to use the collective knowledge to find the optimal solution.

At the initial state, each particle has a random initial position in the search space which is represented by a vector $\vec{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, where $D$ is the dimension of the search space. A particle has a velocity as $\vec{v}_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. The velocity is limited by a threshold $v_{max}$ so that for any $i$ and $d$, $v_{id} \in [-v_{max}, v_{max}]$. During the search process, each particle maintains a record of its best position so far, called the personal best ($pbest$). The best positions among all the personal best positions of its neighbors is the global best ($gbest$). The position and velocity of each particle are updated according to the following equations:

\[
x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1},
\]

\[
v_{id}^{t+1} = \omega \cdot v_{id}^t + c_1 \cdot r_{1i} \cdot (p_{id} - x_{id}^t) + c_2 \cdot r_{2i} \cdot (p_{pd} - x_{id}^t).
\]

Here, $t$ is the index of iteration, $d$ is the index of dimension. The inertia weight $\omega$ is used to balance the local and global search abilities. The parameters $c_1$ and $c_2$ are the acceleration constants. $r_{1i}$ and $r_{2i}$ are random constants following the uniform distribution in the interval $[0, 1]$, $p_{id}$ and $p_{pd}$ denote the values of $pbest$ and $gbest$ in the $d$th dimension of the $i$th particle.

PSO was initiated to solve continuous optimization problems: position update function has been developed for real numbers. To address discrete problems, Kennedy and Eberhart developed a binary PSO [23]. We applied BPSO in WSLAP in previous study [11] and used the linear aggregation (weighted sum) approach to transform the multiple objective values into a single aggregated value. BPSO can provide a single solution and therefore it is used in the scenarios where a service provider knows their budget and required service response time.

Several multi-objective optimization algorithms are based on PSO such as Multi-Objective PSO (MOPSO) [24], and Non-Dominated Sorting PSO (NSPSO) [25]. Coello et al [24] study several multi-objective algorithms, NSGA-II, Micro-GA [26] and MOPSO, and shows that MOPSO is the
most capable of generating the best set of non-dominated solutions close to the true Pareto front with low computational cost. To improve the diversity of non-dominated solutions, Raquel et al. [16] propose an MOPSOCD extended from the MOPSO. They select the global best from an external archive set. The selection is based on crowding distance values (larger the better). MOPSOCD is able to generate a well-distributed set of non-dominated solutions, in this paper, we develop a binary version of MOPSOCD to solve the WSLAP.

3 Problem Description

We consider the WSLAP as a multi-objective problem with two conflicting objectives, minimizing the total deployment cost and minimizing the network latency. This section explains symbols and variables of the problem.

We consider a set of user centers \( U = \{U_1, \ldots, U_m\} \) and a set of candidate locations \( A = \{A_1, \ldots, A_n\} \). A user center denotes the center of a user-centered area which allows latency estimation between user groups and services. Candidate locations are locations that are suitable to deploy web services, e.g., locations of datacenters. A service provider needs to deploy a set of web services \( W = \{W_1, \ldots, W_s\} \). Each service needs to be deployed to at least one location. One can deploy duplicated copies of the same web service to multiple locations in order to reduce the service response time. For each web service \( W_i \) \( i \in W \) and each candidate location \( A_j \in A \), there is a deployment cost \( C_{ij} \) induced by deploying service \( W_i \) to candidate location \( A_j \). \( F_{ki} \) denotes service invocation frequency from user centers \( U_k \) to services \( W_i \) over a unit period of time. In reality, service request frequency patterns may change over time at a steady pace (e.g. in periods of month). Therefore, we consider WSLAP as an off-line problem and assume services’ frequencies are fixed [27]. This study takes an average number of invocations over a period of time to represent the frequencies. This way, the problem is simplified and the research is focused on developing algorithms to search effectively in the complex search space. For each user center \( U_k \in U \) and each candidate location \( A_j \in A \), a latency \( L_{kj} \) represents the response time from the location \( A_j \) to the user center \( U_k \). Because the networking topology is complex, we treat it as a black box. We use a service location matrix \( X = [X_{ij}] \) to represent the location allocation plan, where \( X_{ij} \) represents whether a service \( W_i \) is deployed at a candidate location \( A_j \). WSLAP allocates a set of services \( W = \{W_1, W_2, \ldots, W_s\} \) to a set of candidate locations \( A = \{A_1, A_2, \ldots, A_n\} \) so that the total deployment cost \( f_1 \) and response time \( f_2 \) are minimized. Total deployment cost \( f_1 \) and total network latency \( f_2 \) can be calculated as follows:

\[
f_1 = \sum_{i=1}^{s} \sum_{j=1}^{n} C_{ij} X_{ij}, \quad (5)
\]

\[
f_2 = \sum_{k=1}^{m} \sum_{i=1}^{s} F_{ki} R_{ki}, \quad (6)
\]

where \( X_{ij} \) takes 1 if service \( W_i \) is allocated to location \( A_j \), and 0 otherwise. \( R_{ki} \) stands for the shortest response time of accessing service \( W_i \) from user center \( U_k \), which is calculated as

\[
R_{ki} = \min \{L_{kj} \mid j \in \{1, \ldots, n\} \text{ and } X_{ij} = 1\} \quad (7)
\]

Here, \( R_{ki} \) is represented by the smallest latency \( L_{kj} \) chosen from deployed locations.

WSLAP has the following two objective functions and one constraint.

\[
\text{minimize } f_1 = \sum_{i=1}^{s} \sum_{j=1}^{n} C_{ij} X_{ij}, \quad (8)
\]

\[
\text{minimize } f_2 = \sum_{k=1}^{m} \sum_{i=1}^{s} F_{ki} R_{ki}, \quad (9)
\]

subject to \( \sum_{j=1}^{n} X_{ij} \geq 1, \forall i \in 1, \ldots, s \quad (10) \)

To solve the WSLAP we aim to find an allocation matrix \( X = [X_{ij}] \) such that it results in minimal overall network latency and overall deployment cost.

For example, assume we are given the service invocation frequency matrix \( F \), latency matrix \( L \), deployment cost matrix \( C \), and the allocation matrix \( X \) as follows, we will show how to calculate the overall deployment cost and network latency.

\[
F = U_2 \begin{pmatrix} 150 & 83 & 40 \\ 36 & 42 & 13 \end{pmatrix}, \quad L = U_2 \begin{pmatrix} 0 & 2.8 & 7.6 \\ 2.8 & 0 & 3.1 \end{pmatrix}
\]

\[
C = W_2 \begin{pmatrix} 120 & 95 & 60 \\ 37 & 43 & 65 \end{pmatrix}, \quad X = W_2 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}
\]

The calculation of the overall deployment cost sums up the deployment costs of all the deployments.

\[
f_1 = \sum_{i=1}^{s} \sum_{j=1}^{n} C_{ij} X_{ij} = C_{11} X_{11} + C_{12} X_{12} + C_{13} X_{13} + \ldots + C_{33} X_{33} = 120 + 43 + 27 + 46 = 236
\]

To calculate the overall network latency, we need to first calculate the response time matrix \( R \) using the matrices \( L \) and \( X \). For each service \( W_i \), by checking matrix \( X \), we found out which locations the service has been deployed in. Then, we check matrix \( L \) to find out the corresponding latency from each deployed location \( A_j \) to each user center \( U_k \). Finally, the smallest latency is selected to be the response time from each service \( W_i \) to each user center \( U_k \). In the above example, the corresponding response matrix is

\[
R = U_2 \begin{pmatrix} 0 & 2.8 & 2.8 \\ 2.8 & 0 & 0 \end{pmatrix}
\]

\[
U_3 \begin{pmatrix} 0.8 & 1.2 & 1.2 \end{pmatrix}
\]
Finally, we can calculate the overall network latency as 
\[
f_2 = \sum_{k=1}^{n} \sum_{i=1}^{m} F_{ki} R_{ki} \\
= F_{11} R_{11} + F_{12} R_{12} + F_{13} R_{13} + \ldots + F_{33} R_{33} \\
= 150 \times 0 + 83 \times 2.8 + 40 \times 2.8 + \ldots + 9 \times 1.2 \\
= 591.6
\]

The constraint requires a web service is at least deployed to one location. The example matrix \( X \) above satisfies the constraint.

## 4 BMOPSOCD for Web Service Location Allocation Problem

This section presents the proposed BMOPSOCD. We first define the representation of the problem followed by rounding methods that can be used to transform a continuous representation into a binary form. We then present fitness functions and constraint handling method.

### 4.1 Particle Representation

A solution for WSLAP is an \( s \times n \) allocation matrix \( X = [X_{ij}] \). We need to transform the \( s \times n \) matrix into a vector \( y \) in order to use PSO to generate vector-based solutions. The element \( X_{ij} \) in \( X \) corresponds to the \((n \cdot (i-1) + j)^{th}\) element in \( Y \). Also, for continuous PSO, each element of a particle takes value from 0 to 1, i.e., \( 0 \leq Y_u \leq 1 \). For example, the following \( 3 \times 3 \) matrix
\[
A = \begin{pmatrix}
A_1 & A_2 & A_3 \\
W_1 & 0.12 & 0.87 & 0.42 \\
W_2 & 0.07 & 0.32 & 0.95 \\
W_3 & 0.76 & 0.64 & 0.27
\end{pmatrix}
\]
can be transformed into a vector:
\[
Y = [0.12, 0.87, 0.42, 0.07, 0.32, 0.95, 0.76, 0.64, 0.27].
\]

To transform continuous particle representation to binary form, rounding functions can be applied. The choice of the rounding function plays an important role in the quality of the final results. The following sections discuss different rounding methods. Note that the vector \( Y \) is used in the evolution of PSO. During the fitness evaluation phase, each \( Y \) is first decoded into a matrix \( X \). Then, a rounding function is used to round \( X \) into a binary representation. At last, the evaluation is processed on binary solutions.

### 4.2 Rounding Functions

The original MOPSOCD is designed as a continuous version of PSO. Instead of changing the particle to a binary representation, we still use the continuous representation. This is because the binary position updating function is frequently criticized for being independent between the current value and the next value [28]. However, the position updating of continuous PSO is a well-defined mechanism. Therefore, we keep the continuous representation and related evolutionary operators. Furthermore, we explore using rounding functions as a mechanism of transferring a continuous representation to a binary representation.

At the initial stage, particles are initialized in real values as before. The updates of velocity and position are performed as usual. At the evaluation stage, particles in continuous representation need to be transformed to binary representation, and then to be evaluated by fitness functions. A particle does not change after evaluation; only its “binary representation” is evaluated.

The rounding function maps real values to binary values. The common strategy is to round a real value to its closest integer number. A round-down strategy is adopted in Laskari [29] to solve integer programming problem. Xue [30] defines a static rounding function for feature selection problem. Whether a feature is selected or not is determined by a predefined static threshold \( \theta \). Haupt et al [31] use a real-valued representation of chromosome for GA. Then, a real-valued chromosome is rounded to an integer representation. Nonetheless, there is no thorough study of how to choose rounding functions or a comparison between different rounding functions. Hence, we propose three types of rounding functions and study their effects.

#### 4.2.1 Static Rounding Function

The static rounding function is shown in Equation [11], a parameter threshold \( \theta \) is introduced to round up or round down a particle entry \( X_{ij} \). The threshold value \( \theta \) is ad-hoc and is usually set based on empirical study.
\[
X_{ij} = \begin{cases} 
1 & \text{if } X'_{ij} > \theta \\
0 & \text{otherwise (11)} 
\end{cases}
\]

#### 4.2.2 Dynamic Rounding Functions

The threshold plays an important role in searching for solutions for a given problem [11]. The static rounding function has the following drawbacks. Firstly, it needs to be predefined and problem specific. It is hard to estimate the effect of \( \theta \) before observing results. Secondly, the influence of different threshold values is not completely studied. Therefore, we propose dynamic rounding functions two steps. First, the function adjusts the value of threshold \( \theta \) according to two predefined parameters: a lower bound \( l \) and an upper bound \( u \) of the threshold \( l < u \), and a dynamic parameter: the current generation \( t \). The second step either rounds up or down the value of \( X_{ij} \) according to \( \theta \). Three dynamic rounding functions: Linear function, Quadratic function and Reciprocal function are considered. Specifically, current generation \( t \) cannot equal max_gen in order to keep the result valid.
\[
\theta_{linear} = \frac{l - u}{2} t + u \\
\theta_{quad} = \frac{l - u}{4} t^2 + u \\
\theta_{recip} = u - \frac{u - l}{\text{max_gen}} t (t \neq \text{max_gen})
\]

The reason that we design three dynamic functions is to compare the impact of different trajectories of dynamic thresholds (see Figure [1]). The threshold values with a Linear function are uniformly distributed, whereas Reciprocal and Quadratic functions are unevenly distributed. The performances of these rounding functions are studied in Section 5.5.3.
4.2.3 Stepped Rounding Function

As the dimensionality of the problem increases, the performance of evolutionary computation drops. It is necessary to build a system that can reuse the learned knowledge. Transfer learning is a process to reuse the knowledge in solving unseen tasks [32]. This section proposes a Stepped rounding function that is embodied in the transfer learning process.

Figure 2 shows the evolutionary process with a Stepped rounding function. Initially, the threshold $\theta$ is set to an upper bound $u$ (e.g. 0.7). Then the PSO runs with this setting for a predefined interval of $i$ (e.g. 10) generations. At the beginning of the next interval (e.g. 11) generation, the threshold $\theta$ is changed according to Equation 15 and remains until next interval. This process is repeated until the lower bound $l$ is reached. The optimization is equivalent to initializing a new set of a population with the old one. Therefore, the knowledge is inherited. An underlying assumption is that, if the particle swarm has converged within an interval, then it is better to explore a different direction. Therefore, in the next interval, the swarm will explore a slightly different area and is directed by an adjacent threshold value. The Stepped rounding function is shown in Equation 15 where $\theta'$ denotes the current threshold value.

$$\theta = \begin{cases} 
\theta' - \frac{u - l}{\text{max}_n(i - 1)} & \text{if } (\text{cur}_\text{gen} \mod i) = 0 \\
\theta' & \text{otherwise} 
\end{cases}$$

4.3 Fitness Function and Constraint Handling

After the particles represented as vectors have been transformed to allocation matrices, two fitness functions, Equation 5 and 6, are used to evaluate the fitness of particles that represent allocation plans. Based on the fitness of solutions a set of non-dominated solutions are returned.

As we see in Section 3, WSLAP needs to satisfy the constraint defined by Equation 10 which means that each service needs to be allocated to at least one location. However, during the searching process of PSO, the constraint may be violated. That is, some services might be allocated to none of the locations during the process of searching.

The constraint handling method used by BMOPSOCD is a ranking of violations. A solution $I$ is considered to constraint-dominate a solution $J$ if any of the following conditions is true:

1) Solution $I$ is feasible while solution $J$ is not,
2) Both solutions are infeasible and solution $I$ has fewer violations,
3) Both solutions are feasible, and solution $I$ dominates solution $J$.

The particle with fewer violations is always considered as a better solution. If there is only one constraint, this constraint handling method provides the same effect as the death penalty method in [33].

4.4 The BMOPSOCD algorithm for Web Service Location Allocation Problem

Algorithm 1 presents our BMOPSOCD algorithm for solving the WSLAP. The algorithm iteratively improves results by keeping good solutions and eliminating poor solutions.

First, a population of particles is random initialized from a uniform distribution and evaluated (lines 1–5). Notice that, instead of randomly generating population, we can design an initialization operator that only generates feasible solutions. However, this is not our focus. Therefore, this work only uses random initialization.

Next, non-dominated solutions are stored in the external archive (line 6). Lines 7–20 are the evolution process, where the population evolves maximal $\text{MAX}_t$ generations and $t$ is the current generation. In each generation, solutions are ranked by their crowding distances (lines 8–9). Line 10 randomly selects a global best since they are equally good. Lines 12–13 update position and velocity values. Line 14 guarantees a particle will not leave its domain by reversing its direction. Lines 15 conducts the mutation, where $P_m$ is the mutation rate (see Section 5.6). Line 16–17 round the population and evaluate them with fitness functions. Line 19 updates the best solutions in the archive.

The selection of $p\text{best}$ and $g\text{best}$ is one of the key steps in BMOPSOCD. $p\text{best}$ is the personal best solution of each particle in a population. The $p\text{best}$ is updated only if the new particle dominates the current one. Otherwise, it remains unchanged. In the BMOPSOCD, any non-dominated solutions in the archive can be a $g\text{best}$. Therefore, it is important to ensure that the particles move to an unexplored area. The $g\text{best}$ is selected from non-dominated solutions with the highest crowding distance value. It ensures that the swarm move to a less crowded area.

The fitness of particles can be evaluated using the objective functions presented in Equation 5 and 6. Line 12 updates velocity $v_i$ as:

$$v_{id} = w * v_{id} + c_1 * r_1 * (p_{id} - x_{id}) + c_2 * r_2 * (p_{pg} - x_{id})$$
This study proposes a multi-objective approach to WSLAP to produce well-distributed Pareto-optimal solutions. Meanwhile, the performance of the proposed algorithm performs well with the increasing size of problems. The above section presents our BMOPSOCD approach to WSLAP. We develop static and dynamic rounding methods with different threshold settings. We conduct a set of experiments over three variant of the proposed algorithm. The first variant uses the static threshold. We study the influence of different threshold values; the second variant uses the dynamic rounding functions. We examine the effect of three dynamic rounding functions; the third variant is the Stepped rounding function. We compare its results with dynamic functions. In addition, we make a comparison between a dynamic function with a congregation of several static rounding functions with different thresholds. Lastly, we conduct an experiment considering the overall performance of a BMOPSOCD with a dynamic rounding function in comparison with three other algorithms: PSO, BNSPSO and NSGA-II (see [12]).

All algorithms were implemented in R version 3.3.3 and the experiments were conducted on an i7-4790 3.6 GHz with 8GB of RAM memory running Linux Arch 4.9.6.

5.1 Comparison Design

We use different ways to compare three variants of BMOPSOCD, and BMOPSOCD with other algorithms (e.g NSGA-II). The major difference is that we compare three variants with the solutions from a single run while we compare BMOPSOCD and other algorithms with their best solutions from 40 runs.

Explicitly, for comparing three variants of BMOPSOCD, we first find out the median HyperVolume value (explained in Section 5.3) in 40 runs. Then we find out in which run that the variant of BMOPSOCD produces the median HyperVolume value. Finally, we plot the solutions from that run in terms of two objectives fitness values.

To compare with BMOPSOCD, we first aggregate the solutions from 40 runs and then apply an non-dominated sorting on the aggregated results to find the best results from that algorithm. Finally, we plot the solutions.

All fitness values are normalized between 0 and 1 with a linear normalization method described in [12]. The normalization method requires the maximum and minimum values of cost and response time that we found during optimization.

We obtained these extreme values as follows. For each instance, we collected the solutions from all the four compared algorithms: NSGA-II, BPSO, NSPSO, and BMOPSOCD. Then, we applied non-dominated sorting on these solutions to obtain the non-dominated solutions. Finally, we can find the extreme values of cost and response time from the non-dominated solutions obtained from all the solutions found by all the compared algorithms.

5.2 Test instances

We designed 14 test instances with increasing size of search space (see Table 1). The instances were generated randomly, and thus were not biased to any algorithm. The search space is decided by the number of services and candidate locations (e.g. 2^service\_candidate). Because the computation complexity is not solely determined by the search space of test instances, from instance 3 onwards, for each search space, we vary the number of user centers.

We constructed test instances based on real-world WS-Dream dataset [54]. The dataset contains only the latencies...
between candidate locations and user centers, but lacks the deployment costs for candidate locations and invocation frequencies for web services. Therefore, to generate a complete WSLAP instance, we randomly generated the deployment costs for candidate locations according to a normal distribution with the mean of 100 and standard deviation of 20. We generated the invocation frequencies from a uniform distribution between 1 and 120.

The archive size and a population size are set to 250 and 50 respectively according to the original paper’s recommendation. Generally, the population size has a relation with the search space. A larger population size denotes a stronger search ability because more space are explored. Our study focuses on the binary PSO technique instead of selecting the best parameter set. Therefore an empirical value is selected and the maximal number of iteration is set to 50 in all experiments. The “empirical value” means we tried several values for the number of generation and see whether the solutions has converged (has little changes of fitness values between two consecutive generations). We chose the minimum number of generation that allows all the four compared algorithms converging. Hence, it is fair for all compared algorithms.

We used (1, 1) as the reference point in HyperVolume. Because we normalize two objectives between 0 and 1, reference point (1, 1) is the extreme point of objective values. Moreover, we do not have particular preference to objectives. Therefore, a balanced point (1, 1) is suitable for the reference point.

For each experiment, the proposed algorithm was run for 40 times independently. The best results of all the runs are compared. To obtain the best result of 40 runs, the results of all 40 runs are combined and sorted by the fast non-dominated sorting.

### 5.3 Performance Metrics

We use HyperVolume (HV) and IGD as the evaluation metrics. HyperVolume indicator [18], [35] is a measure used in evolutionary multi-objective optimization, which reflects the volume enclosed by a solution set and a reference point. A larger HyperVolume value indicates a better solution set. The IGD [36] is a modified version of generational distance [37] as a way of estimating how far the elements in the true Pareto front are from those in the non-dominated set produced by an algorithm. IGD calculates the sum of the distances from each point from the true Pareto front to the nearest point from the non-dominated set produced by an algorithm. The lower the IGD, the better quality the solution is. Calculating the IGD value needs a true Pareto front. For our problem, the true Pareto front is unknown. Therefore, an approximated Pareto front is produced by combining all the solutions produced by the four compared algorithms (BMOPSOCD, NSGA-II, BNSPSO, BPSO) and then applying a non-dominated sorting to obtain the final non-dominated set. The approximated Pareto front dominates all the other solutions we found.

### 5.4 Parameter Settings

The parameters of the PSO algorithms are set as follows. We applied static inertia weight and set \( w \) to 0.4. Although it is found in [35] that inertia weight \( w \) decreasing from 0.9 to 0.4 provides excellent results, this work only focuses on rounding functions. Therefore, the influence of dynamic \( w \) has to be eliminated. Mutation probability is in range of [0, 1]. This study follows the original paper’s [16] and set \( P_m \) to 0.5. The cognitive parameter and social parameter \( c_1 \) and \( c_2 \) are generally selected from the range of [0, 4]. This study sets both parameters to 1 [16] so that cognitive and social parameter have an equal influence on the swarm.

### 5.5 Experiments on Rounding Functions

This section designs four experiments to study the effect of different types of rounding functions. We applied Wilcoxon signed rank test on all results between variants of BMOPSOCD. We chose these four representative instances (instances 2 to 5) to show the results. The other test instances showed similar patterns.

#### 5.5.1 Static Rounding Function

There are two questions that we would like to answer with this experiment. The first question is what the influence of the threshold is. The second question is how to select a proper static threshold. We conduct several experiments to evaluate the performance of various threshold values ranging from 0 to 1 divided equally into 10 segments.

#### 5.5.2 Result Analysis of Static Rounding Function

A representative result is given in Figure 3, where each color represents a threshold value and each point denotes a service allocation solution in the 2D objective space. It shows that in the BMOPSOCD with a different static threshold value will lead to solutions covering a different partial region of the Pareto front.

The impact of a static threshold can be explained as a preference factor between two objectives. Take the threshold value of 0.7 as an example, when rounding function...
rounds a real value to a binary value with the threshold of 0.7, it means that 70% of probability rounds this value to zero. It denotes that there are 70% of chances not deploying a web service in a location. Intuitively, the optimization with threshold of 0.7 would consider optimizing cost over latency by deploying fewer web services. The swarm is under the guidance of a "savings" policy.

Another important point is that, the solutions generated using various thresholds, from 0.3 to 0.7, almost cover the entire Pareto front (with the zero cost-fitness eliminated because of the service number constraint). That is, the solutions from thresholds under 0.3 and greater than 0.7 have trivial improvement for a final solution set Therefore, 0.3 and 0.7 can be set as the lower and upper bound of a variant threshold without losing many solutions.

The experimental results answered the first question: the influence of different thresholds. However, it does not offer a guideline for selection of a proper threshold, because none of the solutions could cover the entire Pareto front. The combination of all the results boosts the diversity of the non-dominated set. However, repeating the experiment with different threshold values is time consuming. It is necessary to discover a method with widely spread solutions as well as a low computational complexity. The observation of thresholds inspires the development of the dynamic rounding functions.

5.5.3 Dynamic Rounding Function

In Section 4.2.2, three dynamic rounding functions are proposed. In this section, we evaluate the performances of these three rounding functions to find out which dynamic rounding function provides the best results. The upper bound of dynamic threshold \( u \) is set to 0.7 and the lower bound of dynamic threshold \( l \) is set to 0.3. The results are compared in terms of mean IGD and HV.

5.5.4 Comparison between Dynamic Rounding Functions

Table 3 shows the average performance of the three dynamic functions that are evaluated by HyperVolume. The results indicate that the Reciprocal function outperformed other two dynamic functions in all test cases. This could be visually observed by plotting the best results (Figure 5). Figure 5 shows that the Reciprocal function slightly dominates the other two dynamic functions. However, the problem of the Reciprocal function is that it may miss some regions in the Pareto front, e.g. the regions between dashed-vertical lines of 0.45 and 0.55 in Figure 5. The gap is created because of the lack of uniformity (Figure 1).

The experimental results show that in terms of convergence, the Reciprocal function produces the best result. However, it also shows the disadvantage of the Reciprocal function, which cannot provide a non-dominated set that covers the entire Pareto front. In contrast, the Quadratic function performs slightly worse in convergence but obtains a good-coverage non-dominated set. The Linear function is dominated by Quadratic function in both aspects.

The better convergence with the Reciprocal function can be explained, as there is a minor change in threshold in the most generations and the swarm has a longer time to search along the same direction. However, with the Linear and Quadratic function, the constant changing in direction does not give the algorithm enough time to find a good solution.

Between the Quadratic function and the Linear function, their changing rate of threshold value are different: the Linear function changes equally, while the Quadratic function changes little at the beginning and then become larger. As the swarm searches the low-cost region (as the
threshold starts decreasing from 0.7) and moves to high-cost region, it has more time to search in the low-cost region with the Quadratic function. This is the key reason that the performance of the Quadratic function is better than that of Linear function, because we have observed that the number of solutions in the low-cost region is much smaller than in the high-cost region, as shown in Figure 4, the low-cost regions are quite sparse compared with the low-latency regions. Therefore, it is reasonable to give the algorithm more time to search the low-cost region, while, in the low-latency region, the algorithm can spend much less effort because the solutions are rich. In general, algorithm with the Quadratic function obtains better solutions in the low-cost region than with Linear function.

From table 3, in terms of HyperVolume, the Reciprocal function has the best performance. However, we can observe that the Reciprocal function cannot provide a good coverage of the Pareto front in Figure 5. That is, the Reciprocal function missed many useful solutions because it lacks uniformity. Therefore, we cannot determine which algorithm is better solely depending on the HyperVolume values. We conclude that the Quadratic function performances the best. Although the Quadratic function is worse than the Reciprocal function in terms of HyperVolume value, the disadvantage of Reciprocal function cannot be ignored.

### 5.5.5 Stepped Rounding Function

We study the performance of the Stepped rounding function in this experiment. The upper and lower bound $u$ and $l$ are set to 0.7 and 0.3 respectively. The interval is set to

<table>
<thead>
<tr>
<th>Instance</th>
<th>Reciprocal</th>
<th>Stepped</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.74 ± 0.013</td>
<td>0.72 ± 0.011</td>
</tr>
<tr>
<td>2</td>
<td>0.815 ± 0.013</td>
<td>0.83 ± 0.014</td>
</tr>
<tr>
<td>3</td>
<td>0.87 ± 0.014</td>
<td>0.85 ± 0.014</td>
</tr>
<tr>
<td>4</td>
<td>0.86 ± 0.020</td>
<td>0.85 ± 0.022</td>
</tr>
</tbody>
</table>

Table 4: A comparison between the Reciprocal function and the Stepped function, the mean and standard deviation of HyperVolume values over the 40 independent runs

10 generations. The results are compared with dynamic functions.

### 5.5.6 Results for the Stepped Rounding function

We compared the performance of the Stepped rounding function with the Reciprocal rounding function. The averages of HV of three rounding functions are shown in Table 4. Table 4 shows that in terms of HV, Reciprocal function performs better than the Stepped threshold approach in most cases. However, Figure 6 shows that the Stepped rounding function dominates in instance 3 and has better diversity in instance 4. The results indicate that the Stepped rounding function could provide a uniformly distributed non-dominated set. Overall, the performance of the Stepped rounding function is very close to the Reciprocal function and almost the same with the Quadratic function.

### 5.5.7 Combination of Static Function

In this experiment, we combined all solutions from 5 static rounding functions mentioned in Section 5.5.1 and applied a fast non-dominated sorting over it. The performance is compared with the Reciprocal rounding function in Figure 7. As the figure shows, the combined non-dominated set dominates all the other results in all the test instances.
The solutions are not only diverse but also uniformly distributed. However, the method takes five times longer than using the Reciprocal function.

Regarding execution time (Table 6), BMOPSOCD has similar execution time with other approaches. It achieves the best or the second best performance for 11 out of 14 instances. The reason is the velocity and position calculation (Equation 3 and 4) are linear so that these calculations can be vectorized. Conversely, for binary PSO, new positions are calculated by a sigmoid function which is non-linear. Therefore, an implementation of binary PSO must apply loop in the calculation.

In summary, from the experimental evaluation comparing the proposed algorithm with previous approaches, we observe that on most test instances, BMOPSOCD can achieve much better results than BNSPSO, NSGA-II, and BPSO in both convergence and diversity. The performance of MOPSOCD with dynamic rounding function is not much affected by the number of variables. This is a significant advantage of BMOPSOCD over the other three approaches.

6 RELATED WORK

Location-allocation problems with immobile servers such as hospitals, stores and disposal centers have arisen for a long time [40]. The main characteristic of this problem is that customers need to travel to servers for services in contrast with mobile servers traveling to customers in response to requests. Similar to WSLAP, response time represents customer traveling time and network latency has a strong impact on response time between customers’ locations and web services’ locations.

Most previous researchers treated WSLAP as a single objective problem and proposed many simple greedy-based heuristic algorithms, such as ADD, DROP and Alternate Location Allocation (ALA) [9]. As an improvement, Sun et al. [9] proposed a DAL method which combines DROP, ALA and Linear Programming relaxations of Integer Transportation problems. Aboolian et al. [8] introduced a location-allocation model for a web service provider in duopoly competitive market. They solved the problem with Integer Linear Programming (ILP) technique. Goiri et al. [41] proposed a novel combination of linear programming and simulated annealing approach for allocation of data centers for Internet services. Yuk and Lin [42] considered not only facility location and demand allocation, but also resource capacity allocation (number of service replicas). They formulated the problem as a stochastic mixed integer program and proposed a simulation-based hybrid heuristic to solve the dynamic problem under different response time service level. In contrast, in our previous studies [11], [12], we proposed a multi-objective algorithm with linear aggregation using PSO and a multi-objective algorithm with Pareto front using NSGA-II. Both results show that multi-objective model suits the problem well. The solutions contain a set of equally good allocations trade off between cost and service
response time. Hence, the service providers can select the most suitable allocation according to their preferences.

For example, a multi-objective algorithm gives a solution set containing many solutions, among them, solution A is \{cost: 1000, response time: 3.5s\}, solution B is \{cost: 500, response time: 15s\}, and solution C is \{cost: 800, response time: 10s\}. A single-objective algorithm gives a solution D: \{cost: 900, 12s\}. In this case, solution C better than solution D in both objectives: \{cost: 800 < 900, response: 10s < 12s\}.

Meanwhile, if a service provider would like to provide the best service quality to customers without caring too much of the budget, then solution A is the most suitable choice. Conversely, a service provider with tight budget may choose solution C. Service providers can balance their profit with our multi-objective model.

We can see from the previous example, a service provider can select a suitable solution if the solution set is diverse enough - a provider may have precise requirement (e.g. cost < 525, response < 14.5s). The new approach can provide a much more diverse solution set compared with our previous approaches including BPSO, BNSPSO, and NSGA-II (see Figure. 8). This is a major contribution.

In addition, the major limitation of Integer Linear Programming is that the computational time increases rapidly when the size of the problem increases. Torrent-Fontbona et al. [43] proposed a method that first uses clustering technique to reduce the complexity and then applied heuristic methods to solve the problem. It reduces the computation time as well as the quality of the solutions. However, our proposed algorithms can deal with large test instance with hundreds of services and more. Our BMOPSOCD shows none-degradation with the number of variables increasing.

With the emergence of cloud computing, virtual machine allocation problem becomes important for service providers and cloud providers. Instead of allocating web services to physical servers, web services are first allocated to virtual machines(VMs) and then VMs are allocated to physical machines. The problem of VM allocation is similar to WSLAP with multiple objectives and binary VM allocation. Kessaci et al. [44] proposed an MOGA-CB for minimizing the cost of VMs and response time while considering web service composition as a workflow. Phan et al. [45] proposed a framework called Green Monster, to dynamically move web services across Internet data centers for reducing their carbon footprint while maintaining their performance. Greenmonster applied a modified version of NSGA-II algorithm [19] with an additional local search process. Our BMOPSOCD can solve VM allocation problem with minor modification of objective and constraint functions. Therefore, BMOPSOCD is useful for Cloud providers.

### 7 Conclusion and Future Work

This paper proposed a BMOPSOCD to solve the WSLAP with the aim of producing a set of high-quality solutions that covers most of the Pareto front when dealing with large test instances. For that, we proposed a binary version of multi-objective PSO with crowding distance to solve the WSLAP. We employed a rounding function mechanism which not only makes a continuous algorithm compatible with binary problems but also significantly improves the quality of solutions. Three types of rounding functions were developed. From the experiments, we observed that the
solutions obtained by BMOPSOCD with dynamic rounding functions have a great diversity that almost covers the whole Pareto front. Meanwhile, BMOPSOCD could produce good solutions regardless of the increasing problem size. For service providers who use this algorithm to design the allocation of their services, there are several advantages. First of all, BMOPSOCD can provide near optimum solutions in comparison with solutions from greedy-based heuristic algorithms. Secondly, it provides a wider range of non-dominated solutions compared with single-objective algorithms. In addition, near-optimal solutions are given within a feasible amount of time.

There are a few directions that we can work on in the future. Firstly, our model can be further improved by considering service composition. For now, the problem model considers each service as an atomic service. With the increasing usages of composite services composed with atomic services distributed over the Internet [46], [47], we need to consider service composition workflow while solving WSLAP. Service composition workflows have a significant impact on the allocation of atomic services because the data flow between services could not be neglected. Therefore, the location of each atomic service is highly related to the previous and the next service in a workflow.

Secondly, more potential objectives need to be considered, for example, maximizing availability of services. To
avoid single point failure, web service providers often deploy many copies of service in different candidate locations to ensure the availability.

In addition, service deployment in Cloud becomes a hot topic between service computing and cloud computing. Cloud computing provides an elastic resource management for services so that it removes the burden of managing hardwares for service providers. However, new issues such as dynamic virtual machine placement for service composition, elastic resource management for services have emerged extremely challenging.

ACKNOWLEDGMENT

This work is supported in part by the New Zealand Marsden Fund with the contract numbers (VUW1510 and VUW1614), administered by the Royal Society of New Zealand.

REFERENCES


Boxiong Tan is a Ph.D. student at the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. He received B.E degree from Guangzhou University and MCS degree from Victoria University of Wellington in 2011 and 2016 respectively.

Hui Ma received her B.E. degree from Tongji University and her B.S. (Hons.), M.S. and Ph.D. degrees from Massay University. She is currently a Senior Lecturer in Software Engineering at Victoria University of Wellington. Her research interests include service computing, conceptual modelling, database systems, resource allocation in clouds, and evolutionary computation in combinatorial optimization.

Hui has more than 80 publications, including leading journals and conferences in databases, service computing, evolutionary computation, and conceptual modelling, such as Journal of Heuristics, Data & Knowledge Engineering (DKE), the International Conference on Conceptual Modeling (ER), and Genetic and Evolutionary Computation Conference (GECCO). She has served as a PC member for about 60 international conferences, including seven times as a PC chair for conferences such as ER, DEXA, and APCCM. She is serving as a member of the ER steering committee and the IEEE CIS Emergent Technologies Technical Committee.

Yi Mei (M09) is a lecturer at the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. He received his BSc and PhD degrees from University of Science and Technology of China in 2005 and 2010, respectively. His research interests include evolutionary computation in scheduling, routing and combinatorial optimization, as well as evolutionary machine learning, hyper-heuristics, genetic programming, feature selection and dimensional reduction.

Yi has more than 60 fully refereed publications, including the top journals in EC and Operations Research (OR) such as IEEE TEVC, IEEE Transactions on Cybernetics, European Journal of Operational Research, ACM Transactions on Mathematical Software, and top EC conferences (GECCO). As the sole investigator, he won the 2nd prize of the Competition at IEEE WCCI 2014: Optimization of Problems with Multiple Interdependent Components. He serves as a Vice-Chair of the IEEE CIS Emergent Technologies Technical Committee, and a member of three IEEE CIS Task Forces and two IEEE CIS Technical Committees. He is an Editorial Board Member of International Journal of Bio-Inspired Computation, and a guest editor of a special issue of the Genetic Programming Evolvable Machine journal. He has organised a number of special sessions in international conferences such as IEEE CEC. He serves as a reviewer of over 25 international journals including the top journals in EC and OR.

Mengjie Zhang (M04-SM10) received the B.E. and M.E. degrees from Artificial Intelligence Research Center, Agricultural University of Hebei, Hebei, China, and the Ph.D. degree in computer science from RMIT University, Melbourne, Australia, in 1989, 1992, and 2000, respectively. Since 2000, he has been with the Victoria University of Wellington, Wellington, New Zealand, where he is currently Professor of Computer Science, the Head of the Evolutionary Computation Research Group, and Associate Dean (Research and Innovation) in the Faculty of Engineering. His current research interests include artificial intelligence, machine learning and big data, particularly in evolutionary computation and learning, genetic programming, image analysis, feature selection and big dimensionality reduction, evolutionary scheduling and combinatorial optimization, and evolutionary deep learning and transfer learning. He has published over 450 research papers in refereed international journals and conferences.

Mengjie is a Fellow of Royal Society of New Zealand, and has been serving as an Associated Editor for over ten international journals (including IEEE Transactions on Evolutionary Computation, IEEE Transactions on Cybernetics, Evolutionary Computation Journal, and IEEE Transactions on Emergent Topics in CI) and as a Reviewer of over 30 international journals. He has been serving as a Steering Committee Member and a Program Committee Member for over 100 international conferences. He has supervised over 50 postgraduate research students. He is the Chair of the IEEE CIS Intelligent Systems and Applications Technical Committee, an Immediate Past Chair of the IEEE CIS Evolutionary Computation Technical Committee, a Vice-Chair of the IEEE CIS Task Force on Evolutionary Computer Vision and Image Processing, a Vice-Chair of the IEEE CIS Task Force on Evolutionary Computation for Feature Selection and Construction, a member of IEEE CIS Task Force of Hyper-heuristics, and the Founding Chair for IEEE Computational Intelligence Chapter in New Zealand.