Optimization of Location Allocation of Web Services Using A Modified Non-dominated Sorting Genetic Algorithm

Boxiong Tan, Hui Ma, Mengjie Zhang

School of Engineering and Computer Science,
Victoria University of Wellington, New Zealand
{Boxiong.Tan, Hui.Ma, Mengjie.Zhang}@ecs.vuw.ac.nz

Abstract. In recent years, web services technology is becoming increasingly popular because of the convenience, low cost and capacity to be composed into high-level business processes. The service location-allocation problem for a web service provider is critical and urgent, because some factors such as network latency can make serious effect on the quality of service (QoS). This paper presents a multi-objective optimization algorithm based on NSGA-II to solve the service location-allocation problem. A stimulated experiment is conducted using the WS-DREAM dataset. The results are compared with a single objective genetic algorithm (GA). It shows NSGA-II based algorithm can provide a set of best solutions that outperforms genetic algorithm.

1 Introduction

Web Services are considered as self-contained, self-describing, modular applications that can be published, located, and invoked across the Web [20]. In recent years, web services technology is becoming increasingly popular because of the convenience, low cost and capacity to be composed into high-level business processes [1].

With the ever increasing number of functional similar web services being available on the Internet, the web service providers (WSPs) are trying to improve the quality of service (QoS) to become competitive in the market. QoS, also known as non-functional requirements to web services, is the degree to which a service meets specified requirements or user needs [28], such as response time, security and availability. Among numerous QoS measurements, service response time is a critical factor for many real-time services, e.g. traffic service or finance service. Service response time has two components: transmission time (variable with message size) and network latency [15]. Study [14] shows that network latency is a significant component of service response delay. Ignoring network latency will underestimate response time by more than 80 percent [21], since network latency is related to network topology as well as physical distance [12]. To reduce the network latency WSPs need to allocate services location where has the lower latency to the user center that access the services. Ideally, WSPs could

deploy their services to each user center in order to provide the best quality. However, the more services deployed, the better the quality and the higher cost.

The Web service location-allocation problem is essentially a multi-objective optimization problem [3], for which there are two conflict objectives, to provide optimal QoS to service users and to consume minimal deployment cost. This problem is considered as NP-hard due to the fact that the combinatorial explosion of the search space [22].

Very few researches have studied the service location-allocation problem and most of the researchers treat this problem as a single objective problem. [1] [21] try to solve the problem by using integer linear programming techniques. In particular, [21] solved this problem by employing greedy and linear relaxations of Integer transpotation problem. However, the major problem for this approach is that linear programming is not scaling. Huang [10] proposed an enhanced genetic algorithm (GA)-based approach, which make use of the integer scalarization technique to solve this problem. This algorithm solves the problem with one objective and one constraint. However there has some deficiencies in the integer scalarization techniques [3]. Firstly, the decision maker needs to choose an appropriate weights for the objectives to retrieve a satisfactorily solution. Secondly, non-convex parts of the Pareto set cannot be reached by minimizing convex combinations of the object functions.

So far, to the best of our knowledge, there is no research has considered the service location-allocation problem as a multi-objective problem. Therefore, this paper we will treat service location-allocation problem as a multi-objective problem. Evolutionary multi-objective optimization (EMO) methodologies is ideal for solving multi-objective optimization problems [7], since EMO works with a population of solutions and a simple EMO can be extended to maintain a diverse set of solutions. With an emphasis for moving toward the true Pareto-optimal region, an EMO can be used to find multiple Pareto-optimal solutions in one single simulation run [16]. Among numerous EMO algorithms, Non-dominated sorting GA (NSGA-II) [4], Strength Pareto Evolutionary Algorithm 2 (SPEA-2) [5] have become standard approaches. Some schemes based on particle swarm optimization (PSO) approaches are also proposed [8] [11]. NSGA-II is one of the most widely used methods for generating the Pareto frontier, because it can keep diversity without specifying any additional parameters [6]. In this paper, we propose to use NSGA-II to solve the web service location-allocation problem, which has two objectives: cost and network latency.

We consider the problem faced by a WSP who has existing facilities but wishes to use the collected data to re-allocate their services in order to maximum their profit. The WSP must decide on facility locations from a finite set of possible locations. In order to make a decision, the WSP must first analyze the data collected from current use of services. The collected data should includes the records of invocations from each unique IP address. Therefore, based on these data, the WSP could summarize several customer demands concentrated on n discrete nodes [1], namely user centers. We assume that the WSP has already done this step and a list of user centers and candidate service deployment loca-

tions are given. In addition to deciding locations to of the services, information about network latency between user centers and candidate locations is needed. Exist datasets in [26] [27] contain latency information collected from the real world.

The aim of this project is to propose a NSGA-II based approach to produce a set of near optimal solutions of service location-allocation, so that cost and overall network latency are close to minimum. Then, the service provider could use the algorithm which proposed by this paper, to select an optimal plan based on their funds. The main objectives are:

- To model the web service location-allocation problem so that it can be tackled by NSGA-II.
- To develop a NSGA-II based approach to the web service location-allocation problem.
- To evaluate our proposed approach using some existing datasets.

In Section 2 we introduce the background of NSGA-II and GA. In Section 3 we provide models of the service location allocation problems. Section 4 develops a NSGA-II based algorithm. The experimental design and results evaluation are shown in Section 6. Section 7 provides a brief summary.

2   Background

GA [17] is a powerful tool to solve combinatorial optimization problems. It is an iterative procedure based on a constant-size population. In a GA, a population of strings (called chromosomes or the genotype of the genome), which are encoded as candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem, evolves towards better solutions. Each genome is associated with a fitness value based on a fitness function that indicates how close it comes to meeting the overall specification, when compared to other genomes in the population. The fitness value of an individual is also an indication of its chances of survival and reproduction in the next generation. A typical genetic algorithm requires a genetic representation of the solution domain and a fitness function to evaluate the solution domain. Since a chromosome from the population represents a solution, when the algorithm starts, the whole population moves like one group towards an optimal area so the GA searches from a population of solutions rather than a single solution. Integer scalarization technique [3] is used to solve multi-objective problems with GA. It predefines a weight for each objective.

NSGA-II is a multi-objective algorithm based on GA. When used for problems with only two objectives, NSGA-II performs relatively well in both convergence and computing speed. It permits a remarkable level of flexibility with regard to performance assessment and design specification. NSGA-II assumes that every chromosome in the population has two attributes: a non-domination rank in the population and a local crowding distance in the population. The goal

of NSGA-II is to converge to the Pareto front as possible and with even spread of the solutions on the front by controlling the two attributes.

The algorithm starts with a random initialization population. Once the population is sorted based on non-domination sorting, a rank is assigned to each chromosome. Then, a parameter called crowding distance is calculated for each individual. The crowding distance is a measure of how close an individual is to its neighbors. A large average crowding distance will result in better diversity in the population.

Parents are selected from the population by using tournament selection based on the rank and the crowding distance. An individual is selected in the rank if it is smaller than the other or if the crowding distance is greater than the other. The selected population generates offsprings using crossover and mutation operators.

The population with the current population and current offsprings is sorted again based on non-domination and only the best N individuals are selected, where N is the population size. The selection is based on rank and the on crowding distance on the last front.

3 Problem Description and Modeling

In this section, we first describe the service location-allocation problem, then we will present models for the services location allocation problem.

3.1 Problem Description

Web service location-allocation problem is to determine reasonable locations for web services so that the deployment cost of WSP can be minimized while service performance can be optimized. In this paper, to optimize service performance we consider to minimize network latency.

The task of service location allocation has two objectives:

- To minimize the total cost of the services.
- To minimize the total network latency of the services.

3.2 Model Formulation

To model service location-allocation problem, we need to make use of a set of matrices, to present input information and output solutions.

For service location-allocation problem, we need information of service usage, network latency, and service deployment cost to decide service location-allocation so that the overall network latency can be minimized with minimal deployment cost and within constraints. Assume a set of $S = \{s_1, s_2, ...s_s, s_x\}$ services are requested from a set of locations $I = \{i_1, i_2, ...i_i, i_y\}$. The service providers allocate services to the set of candidate facility locations $J = \{j_1, j_2, ...j_j, j_z\}$.

In this paper, we will use the following matrices.

	Matrices
L	server network latency matrix $L = \{l_{ij}\}$
A	service location matrix $A = \{a_{sj}\}$
F	service invocation frequency matrix $F = \{f_{is}\}$
C	cost matrix $C = \{c_{sj}\}$
R	user response time matrix $R = \{r_{is}\}$

A *service invocation frequency matrix*, $F = [f_{is}]$, is used to record services invocation frequencies from user centers, where f_{is} is an integer that indicates the number of invocations in a period of time from a user center to a service. For example, $f_{13} = 85$ denotes service s_1 is called 85 times in a predefined period of time.

$$F = \begin{matrix} & \begin{matrix} s_1 & s_2 & s_3 \end{matrix} \\ \begin{matrix} i_1 \\ i_2 \\ i_3 \end{matrix} & \begin{bmatrix} 120 & 35 & 56 \\ 14 & 67 & 24 \\ 85 & 25 & 74 \end{bmatrix} \end{matrix} \qquad L = \begin{matrix} & \begin{matrix} j_1 & j_2 & j_3 \end{matrix} \\ \begin{matrix} i_1 \\ i_2 \\ i_3 \end{matrix} & \begin{bmatrix} 0 & 5.776 & 6.984 \\ 5.776 & 0 & 2.035 \\ 0.984 & 1.135 & 2.3 \end{bmatrix} \end{matrix}$$

A *network latency matrix* $L = [l_{ij}]$, is used to record network latencies from user centers to candidate locations. For example, The network latency between user center i_2 with candidate location j_1 is 5.776s. These data could be collected by monitoring network latencies [26] [27].

The cost matrix $C = [c_{sj}]$, is used to record the cost of deployment of services to candidate locations, where c_{sj} is an integer that indicate the cost of the deploying a service to a location. For example, $c_{12} = 80$ denotes the cost of deploying service s_1 to location j_2 is 80 cost units.

$$C = \begin{matrix} & \begin{matrix} j_1 & j_2 & j_3 \end{matrix} \\ \begin{matrix} s_1 \\ s_2 \\ s_3 \end{matrix} & \begin{bmatrix} 130 & 80 & 60 \\ 96 & 52 & 86 \\ 37 & 25 & 54 \end{bmatrix} \end{matrix} \qquad A = \begin{matrix} & \begin{matrix} j_1 & j_2 & j_3 \end{matrix} \\ \begin{matrix} s_1 \\ s_2 \\ s_3 \end{matrix} & \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \end{matrix}$$

To model the service location-allocation problem, we consider the following assumptions:

1. The new WSP decides where to locate his facilities regardless of there is existed functional similar services from other WSPs.
2. The decision of service location-allocation is made only considering two factors: total network latency and total cost.
3. A static allocation policy is used by WSPs. In practice, Web services typically offer clients persistent and interactive services, which often span over multiple sessions. Therefore, a dynamic reallocation scheme is not practical as it may disrupt the continuity of the services.

A *service location-allocation matrix* $A = [a_{sj}]$ represents the actual service location-allocation, where a_{sj} is a binary value 1 or 0 to indicate whether a service is allocate to a location or not.

Using service location allocation matrix $A = [a_{sj}]$ and network latency matrix $L = [l_{ij}]$, we can compute user response time matrix $R = [r_{is}]$,

$$r_{is} = MIN\{l_{ij} \mid j \in \{1, 2, ..., z\} \text{ and } a_{sj} = 1\} \qquad (1)$$

For example, we can use the two example matrices L and A presented above to construct the response time matrix R. For each service s, by checking matrix A, we can find out which location the service has been deployed. Then look back matrix L, find out its corresponding latency to each user center i. If there is more than one location, then the smallest latency is selected. Therefore, we can construct the response time matrix R as:

$$R = \begin{array}{c} \\ i_1 \\ i_2 \\ i_3 \end{array} \begin{array}{ccc} s_1 & s_2 & s_3 \\ \left[\begin{array}{ccc} 5.776 & 6.984 & 0 \\ 0 & 2.035 & 0 \\ 1.135 & 2.3 & 0.984 \end{array} \right] \end{array}$$

4 NSGA-II for Web Services Location Allocation

To apply NSGA-II to the service location-allocation problem, the first step is to define the variable in NSGA-II, i.e. to identify chromosome, genetic operators and the fitness functions.

4.1 Chromosome Representation and Constraints

In our approach, we model the service location matrix $A = [a_{sj}]$ as a chromosome that we mentioned in Section 3.

The constraint setting is based on service providers' needs. One can set multiply constraints to the problem to narrow the potential searching space. In our case, we set two basic constraints. The first constraint service number constraint requires that each service is deployed in at least one location.

$$\sum_{x \in S} a_{xj} \geq 1 \tag{2}$$

The second constraint, is cost constraint, which sets up the upper boundary of the total cost. An integer number $CostLimitation$ is decided by the WSP.

$$\sum_{s \in S} \sum_{j \in J} c_{sj} \times a_{sj} \leq CostLimitation \tag{3}$$

4.2 Genetic Operators

The original NSGA-II uses a simulated binary crossover (SBX) [2] and polynomial mutation [19] to cope with continuous problems. However, our problem is discretized, therefore we use the regular GA mutation and crossover operations.

The selection operator is the tournament selection [23], which allows the highest probability of being reproduced to next generation.

Mutation The mutation operator works as follows: initially choose one random location from the chromosome. Then, this location is replaced by by its inverse value e.g 1 to 0 or 0 to 1.

In this study, a chromosome in a population will be selected based on mutation possibility P_m, which is a parameter predefined and fixed.

$$
\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & \boxed{1} & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{\text{Mutation}} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & \boxed{0} & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}
$$

Fig. 1

Crossover The crossover operator in this paper is the single point crossover. The crossover is controlled by crossover probability P_c, is a parameter predefined and fixed. The crossover point is created randomly within the length of the chromosome.
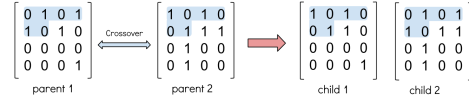
$$
\underset{\text{parent 1}}{\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \xleftrightarrow{\text{Crossover}} \underset{\text{parent 2}}{\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}} \Longrightarrow \underset{\text{child 1}}{\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \underset{\text{child 2}}{\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}
$$

Fig. 2

Fitness Function

In order to accomplish these two objectives, we design two fitness functions to evaluate how good each chromosome meets the objectives. We use CostFitness to calculate the overall cost of deploying services under an allocation plan

$$
CostFitness = \sum_{s \in S} \sum_{j \in J} c_{sj} \times a_{sj} \tag{4}
$$

c_{sj} is the cost of deploying service s at location j, where a_{sj} represents the deployment plan. The sum of the multiplication of c_{sj} and a_{sj} is the total deployment cost.

We assume the latency is symmetrical between user center and candidate location. e.g. The latency from location i to location j is same with the latency from location j to location i.

$$LatencyFitness = \sum_{i \in I} \sum_{s \in S} r_{is} \times f_{is} \qquad (5)$$

Normalise function To indicate the goodness of an allocation solution we normalise CostFitness and LatencyFitness according to the largest and minimum values of CostFitness and LatencyFitness. Normalised fitness values can also be used to compare results from different approaches. Since the maximum and minimum value for total cost and total latency are deterministic, we use exhaustive search to find out the $Latency_{max}$. $Latency_{min}$ is zero for we assume each service could be deployed in each user center. $Cost_{min}$ is the cost of allocation each of services at a location that leads to minimal cost and $Cost_{max}$ is the cost of allocation when each service is allocated to all the locations.

$$CostFitness' = \frac{CostFitness - Cost_{min}}{Cost_{max} - Cost_{min}} \qquad (6)$$

$$LatencyFitness' = \frac{LatencyFitness - Latency_{min}}{Latency_{max} - Latency_{min}} \qquad (7)$$

4.3 NSGA-II based algorithm for service location-allocation

There are two major differences between standard NSGA-II and modified version. Firstly, in order to avoid repeatly evaluate the fitness of chromosome. After the first generation is initialized, the Pareto front will be stored in the memory. In each generation, when evaluate the chromosome, the chromosome is checked to see it exists in the memory pool. If so, then the calculation of fitness will be skipped. At the end of each iteration, the Pareto front pool is updated to the current Pareto front. The reason for setting a memory pool is that, after a number of iterations, the population start converging. Most of the population are redundant. Therefore, a memory pool could considerably reduce the repeated computation.

The other difference is that, we use general mutation and crossover operation instead of polynomial mutation and simulated binary crossover. It is important to note that the mutation and crossover operators can produce solutions that might violate the constraints. Therefore, repair operators are needed to try to maintain feasible solutions. The proposed algorithm checks the cost and service number constraint to avoid possible infeasible solutions.

5　GA for Web Service Location Allocation

In order to show the advantages of our multi-objective NSGA-II approach, we extend the single-objective algorithm GA based approach in [11] to consider two objectives. The extended GA-based approach is show as following. A weight w needs to be predefined in GA. The weight is used for measuring the importance of objectives. Conventionally, the weight is in the range of $[0, 1]$. In our approach, we define the weight equals 0.5.

Algorithm 1 NSGA-II for service location-allocation

Inputs: Cost Matrix C, Server network latency matrix L, Service location matrix A, Service invocation frequency matrix F

Outputs: Pareto Front:a set of service allocation matrix A

1: Initialize a population of chromosome with random binary values
2: Evaluate population with fitness functions
3: Non-dominated sort and assign a ranking to each chromosome
4: Evaluate the Crowding distance of each chromosome
5: Initialize the Pareto Front Pool
6: **while** predefined generation **do**
7:      Apply Tournament Selection
8:      Apply Crossover
9:      Apply Mutation
10:     **for** (**do** each chromosome)
11:         **while** violate service number constraint **do**
12:             random choose a location j and set $a_{sj} = 1$
13:         **end while**
14:         **while** violate cost constraint **do**
15:             random choose a location j and set $a_{sj} = 0$, as long as $\sum\limits_{s \in S} a_{sj} \geq 1$
16:         **end while**
17:         **if** chromosome does not exist in the Pareto front Pool **then**
18:             Evaluate with the fitness functions
19:         **end if**
20:     **end for**
21:     Non-dominated sort and assign ranking
22:     Evaluate the Crowding distance
23:     Recombination and Selection
24:     Update the Pareto Front Pool with the current Pareto Front
25: **end while**
26: Return the Pareto Front

Fitness function As in Section 3.2 we model an allocation matrix as a chromosome. Crossover and mutation operators are same as defined in Section 4.2. To evaluate the chromosomes of population. We use Integer Scalarization technique to calculate the fitness value. w is a predefined value to measure the important of cost and latency. Note that CostFitness and LatencyFitness are calculate in the same as in section 4.2.

$$Fitness = w \times CostFitness + (1 - w) \times LatencyFitness \qquad (8)$$

6   Experiment Evaluation

To evaluate the effectiveness and efficiency of our proposed NSGA-II based approach to service location-allocation. We compare our approach with a GA-based single objective approach using an existing dataset, WS-DREAM [26] [27], which

Algorithm 2 GA for service location-allocation

Inputs: Cost Matrix C, Server network latency matrix L, Service location matrix A, Service invocation frequency matrix F, weight parameter w

Outputs: One best solution: allocation matrix A

1: Initialize a population of chromosome with random binary value
2: Evaluate population with the fitness function
3: **while** predefined generation **do**
4:     Apply Tournament Selection
5:     Apply Crossover
6:     Apply Mutation
7:     **for** (**do** each chromosome)
8:         **while** violate service number constraint **do**
9:             random choose a location j and set $a_{sj} = 1$
10:         **end while**
11:         **while** violate cost constraint **do**
12:             random choose a location j and set $a_{sj} = 0$, as long as $\sum_{s \in S} a_{sj} \geq 1$
13:         **end while**
14:     Evaluate each chromosome with the fitness function
15:     **end for**
16:     Update population
17: **end while**
18: Return population

---

is a historical dataset on QoS of Web services from different locations. It contains the data of latency from 339 different user locations invoked 7824 web services scattered over different locations.

The algorithm was coded in R [18] using existed package: NSGA2R. The program was run on a 3.40GHz desktop computer with 8 GB RAM.

4 different service location-allocation problems are designed with different complexities.

Table 1: Complexity table

problem	user location	server location	number of service
1	3	3	3
2	5	5	5
3	10	10	10
4	15	15	15

A cost matrix is randomly generated from a normal distribution with mean = 100 and standard deviation = 20, in addition, a frequency matrix was an integer matrix which is randomly generated from a uniform distribution on [1, 120].

In each dataset, algorithms run under four different levels of cost constraints: Sufficient condition is 100% the expected total cost, good condition (70%), pool condition (40%) and minimum budget condition (0%). We try to stimulated a real world budget condition with these four scenarios. In the minimum budget condition, both algorithms exhaustively reduce the cost until it reaches the service number constraint. The NSGA-II runs 40 independent times with different random seeds ranging from 1 to 40. To test the efficiency of the algorithms, We evaluate the average run time for each algorithm.

Parameter settings for the algorithms are as follow. The population size is 50, and the maximum number of generations is 50. The tournament size is 3. The crossover probability P_c is 0.8 and the mutation probability P_m is 0.2 as we found that this combination can produce good results. We use same parameter settings for GA.

To compare with the result of NSGA-II and GA, we first derive the Pareto front by using the approach in [24] [25], and then compare the results using approach in [13]. In Xue's approach, NSGA-II generates 40 results (from 40 runs) under each cost constraint are presented in the Section 6.1. 40 sets of solutions achieved by each multi-objective algorithm are firstly combined into one union set. GA also generates 40 sets of solutions, we select the best one based on its fitness value from each set and combine them into one set. The non-dominated solutions are presented to compare with the solutions achieved by GA.

In addition to the comparison between NSGA-II and GA, we conducted full experiments on NSGA-II with an initialisation and GA with an initialisation. We initialized the population with 49 random chromosomes and a chromosome which leading to minimum cost. We expect the initialized algorithms superior than the uninitialized algorithms.

6.1   Effectiveness comparison

We conducted experiments on NSGA-II, GA, NSGA-II with initialisation and GA with initialisation respectively.

We use cost fitness value and latency fitness value as x, y coordinate. Our goal is to minimize both cost and latency. Therefore, better solution should locate close to the origin.

Experimental results show that few different cost constraints leads to similar result patterns. Due to the page limitation we shows the results of one cost constraints.

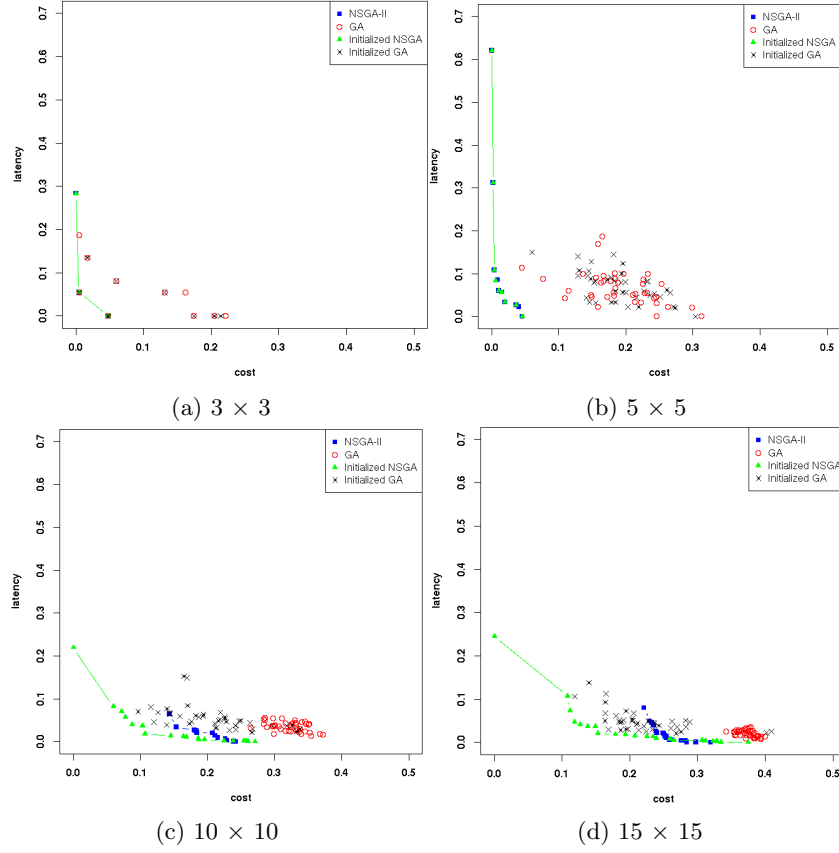(a) 3 × 3

(b) 5 × 5

(c) 10 × 10

(d) 15 × 15

Fig. 3

From the above results we can see that for all the four problems, NSGA-II based approach produces results that dominate or overlap the results from GA based approach. Further, NSGA-II with an initialized chromosome that represents service location-allocation of the minimum cost dominate the results without a chromosome of the lower cost, though problem 1 and problem 2 of small complexity size, this observation is not obvious.

In particular, for big problems 3 and 4, results of NSGA-II based approaches dominate the results of GA-based approaches. We also notice that for problems even though the population size is small as 50, including a chromosome of optimal cost can help to narrow down searching space and to converge to optimal solution faster.

6.2 Efficiency comparison

The results from initialized algorithms are similar with uninitialized algorithms, therefore we only present the uninitialized results. As shown in the table above

Table 2: Efficiency Test

	3 × 3		5 × 5		10 × 10		15 × 15	
	NSGA-II(s)	GA(s)	NSGA-II(s)	GA(s)	NSGA-II(s)	GA(s)	NSGA-II(s)	GA(s)
Sufficient	4.4 ± 0.3	1.6 ± 0.1 ↓	5.9 ± 0.1	3.2 ± 0.1 ↓	13.7 ± 0.1	11.0 ± 0.1	27.0 ± 0.1	23.9 ±0.5↓
Good	4.4 ± 0.2	1.6 ±0.1 ↓	6.0 ± 0.1	3.2 ± 0.1 ↓	13.9 ± 0.08	11.1 ± 0.3	27.2 ± 0.1	24.1 ± 0.27 ↓
Poor	4.6 ± 0.19	2.2 ±0.2 ↓	6.3 ± 0.07	4.29 ± 0.17 ↓	15.2 ± 0.16	14.8 ± 0.3	31.3 ± 0.28↓	33.6 ± 0.45
Minimum	4.6 ± 0.1	2.2 ± 0.1 ↓	7.2 ± 0.12	5.75 ±0.17 ↓	24.12 ± 0.5 ↓	25.8 ±0.5	56.72 ± 1.6 ↓	66.8±1.2

for small problems GA based approach are faster. However for bigger problem (problem 3 and 4) NSGA-II shows its efficiency. Also NSGA-II produces a set of non-dominated solutions instead of one solution, which provide WSPs with more options.

7 Conclusion

In this paper, we proposed a NSGA-II based approach to web service location-allocation problem. Our approach consider two objectives, minimizing cost and minimizing network latency at the same time. We have conducted a full experimental evaluation using the public WS-DREAM dataset to compare our approach to single-objective GA-based approach. The experimental results shows the NSGA-II is effective to produce a set near-optima solutions for the web service location-allocation problem.

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. Beyer, H.G., Deb, K.: On self-adaptive features in real-parameter evolutionary algorithms. Evolutionary Computation, IEEE Transactions on 5(3), 250–270 (2001)
3. Caramia, M.: Multi-objective optimization. In: Multi-objective Management in Freight Logistics, pp. 11–36. Springer London (2008)
4. 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)
5. Deb, K., Mohan, M., Mishra, S.: Evaluating the epsilon-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. Evol. Comput. 13(4), 501–525 (2005)
6. Deb, K., Sundar, J., N, U.B.R., Chaudhuri, S.: Reference point based multi-objective optimization using evolutionary algorithms. In: International Journal of Computational Intelligence Research. pp. 635–642. Springer-Verlag (2006)
7. Desai, S., Bahadure, S., Kazi, F., Singh, N.: Article: Multi-objective constrained optimization using discrete mechanics and nsga-ii approach. International Journal of Computer Applications 57(20), 14–20 (2012), full text available
8. Elhossini, A., Areibi, S., Dony, R.: Strength pareto particle swarm optimization and hybrid ea-pso for multi-objective optimization. Evol. Comput. 18(1), 127–156 (2010)

9. Goiri, I., Le, K., Guitart, J., Torres, J., Bianchini, R.: Intelligent placement of datacenters for internet services. In: Distributed Computing Systems (ICDCS), 2011 31st International Conference on. pp. 131–142 (2011)

10. 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)

11. Huang, V.L., Suganthan, P.N., Liang, J.J.: Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems: Research articles. Int. J. Intell. Syst. 21(2), 209–226 (2006)

12. Huffaker, B., Fomenkov, M., Plummer, D., Moore, D., claffy, k.: Distance Metrics in the Internet. In: IEEE International Telecommunications Symposium (ITS). pp. 200–202. IEEE, Brazil (Sep 2002)

13. Ishibuchi, H., Nojima, Y., Doi, T.: Comparison between single-objective and multi-objective genetic algorithms: Performance comparison and performance measures. In: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on. pp. 1143–1150 (2006)

14. Jamin, S., Jin, C., Kurc, A., Raz, D., Shavitt, Y.: Constrained mirror placement on the internet. In: INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. vol. 1, pp. 31–40 vol.1 (2001)

15. Johansson, J.M.: On the impact of network latency on distributed systems design. Inf. Technol. and Management 1(3), 183–194 (2000)

16. Kanagarajan, D., Karthikeyan, R., Palanikumar, K., Davim, J.: Optimization of electrical discharge machining characteristics of wc/co composites using non-dominated sorting genetic algorithm (nsga-ii). The International Journal of Advanced Manufacturing Technology 36(11-12), 1124–1132 (2008)

17. Man, K.F., Tang, K.S., Kwong, S.: Genetic algorithms: concepts and applications. IEEE Transactions on Industrial Electronics 43(5), 519–534 (1996)

18. Morandat, F., Hill, B., Osvald, L., Vitek, J.: Evaluating the design of the r language: Objects and functions for data analysis. In: Proceedings of the 26th European Conference on Object-Oriented Programming. pp. 104–131. ECOOP'12, Springer-Verlag (2012)

19. Raghuwanshi, M.M., Kakde, O.G.: Survey on multiobjective evolutionary and real coded genetic algorithms. In: Proceedings of the 8th Asia Pacific symposium on intelligent and evolutionary systems. pp. 150–161 (2004)

20. Ran, S.: A model for web services discovery with qos. SIGecom Exch. 4(1), 1–10 (2003)

21. Sun, Y., Koehler, G.J.: A location model for a web service intermediary. Decis. Support Syst. 42(1), 221–236 (2006)

22. Vanrompay, Y., Rigole, P., Berbers, Y.: Genetic algorithm-based optimization of service composition and deployment. In: Proceedings of the 3rd International Workshop on Services Integration in Pervasive Environments. pp. 13–18. SIPE '08, ACM (2008)

23. Xie, H., Zhang, M., Andreae, P., Johnson, M.: An analysis of multi-sampled issue and no-replacement tournament selection. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation. pp. 1323–1330. GECCO '08, ACM (2008)

24. Xue, B., Zhang, M., Browne, W.N.: Multi-objective particle swarm optimisation (pso) for feature selection. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation. pp. 81–88. GECCO '12, ACM (2012)

25. Xue, B., Zhang, M., Browne, W.: Particle swarm optimization for feature selection in classification: A multi-objective approach. Cybernetics, IEEE Transactions on 43(6), 1656–1671 (Dec 2013)
26. 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)
27. 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)
28. 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)