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]. 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. Service response time is a critical measurement in QoS. It has two components: transmission time and network latency [14]. Study [13] 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]. To reduce the network latency WSPs need to allocate services to a location where has the lower latency to the user center that access the services. User center denotes a geometric location (e.g., a city) that is encompassed by a service area. Ideally, WSPs could deploy their services to each user center in order to provide the best quality. However, the more services deployed, the higher deployment cost will be.

The Web service location-allocation problem is essentially a multi-objective optimization problem [2], for which there are two conflict objectives, to provide optimal QoS to Web service users and to consume minimal deployment cost. This problem can be classified as a multidimensional knapsack problem (MKP). Therefore, it is considered NP-hard due to the fact that the combinatorial explosion of the search space [22].

[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 transportation problem. However, integer programming (IP) is very effective for small-scale or mid-scale MKP but suffers from large memory requirement for large-scale MKP [11]. Huang [9] proposes an enhanced genetic algorithm (GA)-based approach, which makes use of the integer scalarization technique to solve this problem. This algorithm solves the problem with one objective and one constraint. However there are some deficiencies in the integer scalarization techniques [2]. Firstly, decision makers need to choose 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.

Evolutionary multi-objective optimization (EMO) methodologies is ideal for solving multi-objective optimization problems [6], 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 [15]. Among numerous EMO algorithms, Non-dominated sorting GA (NSGA-II) [3], Strength Pareto Evolutionary Algorithm 2 (SPEA-2) [4] have become standard approaches. NSGA-II is one of the most widely used methods for generating the Pareto front, because it can keep diversity without specifying any additional parameters [5]. In this paper, we propose to use NSGA-II to solve the Web service location-allocation problem, which has two objectives, to minimize cost and network latency.

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 WSPs could use the algorithm which is proposed by this paper, to select an optimal plan based on their fund constraints. 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 method to solve combinatorial optimization problems. It is an iterative procedure based on a constant-size population. In GA, a population of strings (called chromosomes), which are encoded as candidate solutions (called individuals) 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 meets the overall specification, when compared with 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. Integer scalarization technique [2] is used to solve multi-objective problems with GA, by predefining a weight for each objective.

NSGA-II is a multi-objective algorithm based on GA. When it is used for problems with two or three objectives, NSGA-II performs well in both convergence and computing speed. NSGA-II permits a remarkable level of flexibility with regard to performance assessment and design specification. It 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 much as possible and with even spread of the solutions on the front by controlling the two attributes.

3 Problem Description and Modeling

3.1 Problem Description and Assumptions

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.

In the mean time, service providers have cost constraints which limit the total cost of services deployment.

Stakeholder Web Service Providers Assume the historical information of Web service usage has been collected. WSPs wish to allocate services to servers in candidate locations in order to maximum their profit.

The WSP must decide on services locations from a finite set of possible locations. In order to make a decision, the WSP must obtain data of service usages. 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 locations are given. A candidate location is the geometric location that is suitable to deploy services. User centers and candidate locations are very likely overlapping when Web serivce are deployed locally to user centers. In addition to deciding locations of the services, information about network latency between user centers and candidate locations are needed.

The list below shows some critical information that should be provided by the WSPs. 1. A list of user centers. 2. A list of candidate locations 3. Service

invocation frequencies from user centers to services 4. Average network latencies from user centers to candidate locations 5. Web service deployment cost for each candidate location

These are the main input data that the decision making is dependent on. The details of these input data and modeling are introduced in Section 3.2. Worth noting that service invocation frequencies are changing over time. Network latency highly depends on the network traffic and may be very different during periods of a day. However, as long as there is no significant changes in the network topology, the average network latency remain stable. Therefore, the average network latency for a period of time should be representative.

Although dynamic service deployment is possible [16], the static deployment is still the mainstream [8]. In this paper, we made an assumption that WSPs periodically change the Web service deployment.

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.

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 a set of candidate facility locations $J = \{j_1, j_2, ...j_j, j_z\}$.

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 i to a service s. For example, $f_{31} = 85$ denotes service s_1 is called 85 times in a predefined period of time from user center i_3.

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

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 indicates the cost of 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.

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

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 $R = [r_{is}]$,

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

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 we check matrix L, to 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:

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

4 NSGA-II for Web Services Location Allocation

4.1 Chromosome Representation and Constraints

In our approach, we model the service location matrix $A = [a_{sj}]$ as a chromosome. The constraint setting is based on service providers' needs. In our case, we need set two constraints. The first constraint, *service number constraints*, requires that each service is deployed in at least one location.

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

The second constraint, *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

Our problem is discretized, therefore we use the binary GA mutation and crossover operations [18]. The selection operator is the tournament selection [23], which allows the highest probability of being reproduced to next generation.

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

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

where c_{sj} is the cost of deploying service s at location j, 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., $l_{ij} = l_{ji}$. We use *Latency Fitness* to calculate overall latency of all service request over a period of time.

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

Where r_{is} denotes the minimum network latency from user center i to service s. f_{is} denotes the invocation frequency from i to s

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 values 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 allocating each of services at a location that leads to the minimal cost and $Cost_{max}$ is the cost of allocating each service to all the locations.

$$CostFitness' = \frac{CostFitness - Cost_{min}}{Cost_{max} - Cost_{min}} \tag{6}$$

$$LatencyFitness' = \frac{LatencyFitness - Latency_{min}}{Latency_{max} - Latency_{min}} \tag{7}$$

4.3 NSGA-II based algorithm for service location-allocation

In this section we present our NSGA-II based algorithm for service location-allocation as Algorithm 1, comparing with the original NSGA-II our proposed algorithm has two new features.

Firstly, in order to avoid repeatedly evaluating the fitness of chromosomes, after the first generation is initialized, it store the Pareto front in the memory. In each generation, when evaluate the chromosomes, the chromosomes are checked to see they exist 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.

Secondly, it uses 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 maintain feasible solutions. The proposed algorithm checks the cost and service number constraint to avoid possible infeasible solutions.

Thirdly, we include a solution that leads to minimal cost as an individual in the initialized generation. To do that we expect that it could accelerate the convergence as well as keep the solutions diversity.

5 GA for Web Service Location Allocation

In order to show the performance of our multi-objective NSGA-II based approach, we extend the single-objective GA based approach in [10] to consider

Algorithm 1 NSGA-II for service location-allocation

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

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

Initialize a population of chromosome with random binary values and include a chromosome represents location with minimal cost
Evaluate population with fitness functions
Non-dominated sort and assign a ranking to each chromosome
Evaluate the Crowding distance of each chromosome
Initialize the Pareto Front Pool
while predefined generation **do**
    Apply Tournament Selection
    Apply Crossover
    Apply Mutation
    for (**do** each chromosome)
        while violate service number constraint **do**
            random choose a location j and set $a_{sj} = 1$
        end while
        while violate cost constraint **do**
            random choose a location j and set $a_{sj} = 0$, as long as $\sum_{s \in S} a_{sj} \geq 1$
        end while
        if chromosome does not exist in the Pareto front Pool **then**
            Evaluate with the fitness functions
        end if
    end for
    Non-dominated sort and assign ranking
    Evaluate the Crowding distance
    Recombination and Selection
    Update the Pareto Front Pool with the current Pareto Front
end while
Return the Pareto Front

two objectives. We employ integer scalarization technique [7] to transform the multi-objective problem into a single objective problem. A weight w needs to be predefined in GA. The weight measures the importance of objectives. Therefore, it is used to balance which objective is more favourable to the service provider. Conventionally, the weight is in the range of $[0, 1]$. For example, if we define the weight equals 0.7. It denotes that we consider cost outweigh network latency. In our approach, we define the weight equals 0.5 since we consider both objectives equally important.

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 [7] to calculate the fitness value.

$$Fitness = w \times CostFitness' + (1 - w) \times LatencyFitness' \tag{8}$$

w is a predefined value used to measure the important of cost and latency. Note that *CostFitness* and *LatencyFitness* are calculated using Formula 6 and 7 in section 4.2.

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 the GA-based single objective approach in Section 5 using an existing dataset, WS-DREAM [26] [27], which is a historical dataset on QoS of Web services from different locations. It contains the data of latencies from 339 different user locations invoked 5824 Web services scattered over different locations. The algorithm was coded in R [19] using existed package: NSGA2R. The program was run on a 3.40GHz desktop computer with 8 GB RAM. Four different service location-allocation problems are designed with different complexities. A cost matrix is

Table 1: Test Cases

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

randomly generated from a normal distribution with mean as 100 and standard deviation as 20. In addition, a frequency matrix, is randomly generated from a uniform distribution over [1, 120].

In each dataset, algorithms are run under four different levels of cost constraints: Sufficient condition (indicating services were allocated to all candidate locations), 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 based algorithm is 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 the result of Algorithm 1 with GA-based algorithm, we first derive the Pareto front by using the approach in [24] [25], and then compare the results using approach in [12]. Under each of cost constraint, our NSGA-II based algorithm was run 40 times to generate 40 sets of solutions, which are then com-

bined into one set. Then we applied non-dominated sort over this set. In the mean time, GA may also run 40 times to 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 based algorithm and GA based algorithm, we conducted full experiments on NSGA-II without minimal cost initialisation and GA without minimal cost initialisation. 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 coordinates. Our goal is to minimize both cost and latency. Therefore, better solution should locate close to the origin.

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

From the above results we can see that for all the four problems, NSGA-II based approach produce 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 for problem 1 and problem 2 of small complexity size, this observation is not obvious.

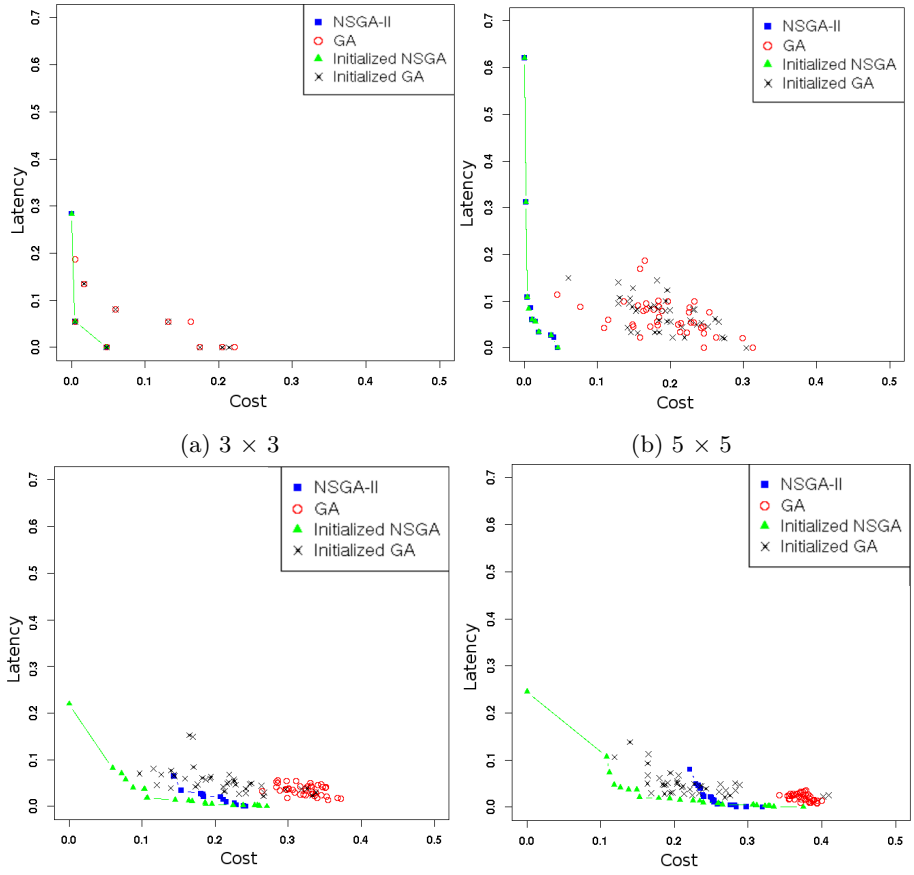
In particular, for big problems, problem 3 and 4, results of NSGA-II based approaches dominate the results of GA-based approaches. We also notice that even though the population size is small as 50, including a chromosome of optimal cost can help to narrow down searching space and to converge to optimal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

6.2 Efficiency comparison

Table 2: Execution Time (s)

	problem 1		problem 2		problem 3		problem 4	
	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

The results from initialized algorithms are similar with uninitialized algorithms, therefore we only present the uninitialized results. As shown in the table above for small problems GA based approach are faster. However for bigger problem (problem 3 and 4) NSGA-II based approach, are more efficient than GA-based approach. Also NSGA-II based approach produces a set of non-dominated solutions instead of one solution, which provide WSPs with more options.



(a) 3 × 3 (b) 5 × 5

(c) 10 × 10 (d) 15 × 15

Fig. 1: Comparisons Between NSGA-II, GA, initialized-NSGA-II and initialized-GA

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 based approach is effective to produce a set near-optima solutions for the Web service location-allocation problem. Also, NSGA-II based approach are more efficient than GA-based approach for problem with big number of user centers and server locations. Future work will investigate the scalability of our proposed approaches for big datasets.

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. Caramia, M.: Multi-objective optimization. In: Multi-objective Management in Freight Logistics, pp. 11–36. Springer London (2008)
3. 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)
4. Deb, K., Mohan, M., Mishra, S.: Evaluating the epsilon-domination based multiobjective evolutionary algorithm for a quick computation of pareto-optimal solutions. Evol. Comput. 13(4), 501–525 (2005)
5. Deb, K., Sundar, J., N, U.B.R., Chaudhuri, S.: Reference point based multiobjective optimization using evolutionary algorithms. In: International Journal of Computational Intelligence Research. pp. 635–642 (2006)
6. 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
7. Ehrgott, M.: A discussion of scalarization techniques for multiple objective integer programming. Annals of Operations Research 147(1), 343–360 (2006)
8. He, K., Fisher, A., Wang, L., Gember, A., Akella, A., Ristenpart, T.: Next stop, the cloud: Understanding modern web service deployment in ec2 and azure. In: Proceedings of the 2013 Conference on Internet Measurement Conference. pp. 177–190. IMC '13, ACM (2013)
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. 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)
11. Hwang, J., Park, S., Kong, I.Y.: An integer programming-based local search for large-scale maximal covering problems. International Journal on Computer Science and Engineering pp. 837–843 (2011)
12. Ishibuchi, H., Nojima, Y., Doi, T.: Comparison between single-objective and multiobjective genetic algorithms: Performance comparison and performance measures. In: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on. pp. 1143–1150 (2006)
13. 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)
14. Johansson, J.M.: On the impact of network latency on distributed systems design. Inf. Technol. and Management 1(3), 183–194 (2000)
15. Kanagarajan, D., Karthikeyan, R., Palanikumar, K., Davim, J.: Optimization of electrical discharge machining characteristics of wc/co composites using nondominated sorting genetic algorithm (NSGA-II). The International Journal of Advanced Manufacturing Technology 36(11-12), 1124–1132 (2008)
16. Kemps-Snijders, M., Brouwer, M., Kunst, J.P., Visser, T.: Dynamic web service deployment in a cloud environment (2012)

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. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press (1998)
19. 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 (2012)
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 (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)