VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Multi-Objective PSO for Service Location-Allocation

Boxiong Tan

Supervisors: Hui Ma, Mengjie Zhang

Submitted in partial fulfilment of the requirements for
Master of Computer Science.

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 impact on the quality of service (QoS). This paper presents a multi-objective optimization algorithm based on multi-objective Particle Swarm Optimization (PSO) to solve the service location-allocation problem. A stimulated experiment is conducted using the WS-DREAM dataset. The results are compared with a multi-objective genetic algorithm (NSGA-II). It shows multi-objective PSO based algorithm can provide a set of best solutions that outperforms NSGA-II.

Contents

Chapter 1

Introduction

Modern enterprises need to respond effectively and quickly to opportunities in today's competitive and global markets. To accommodate business agility, companies are supposed to utilise existing business processes while exposing the various packaged applications found spread throughout the enterprise in a highly standardized manner. A contemporary approach for addressing these critical issues is embodied by (Web) services that can be easily assembled to form a collection of autonomous and loosely coupled business processes [21].

The emergence of Web services developments and standards in support of automated business integration has driven major technological advances in the integration software space, most notably, the service oriented archiecture (SOA) [5]. In an SOA, software resources are packaged as Web services, which are well defined, self-contained modules that provide standard business functionality and are independent of the state or context of other services [22].

A Web service is a software system allowing to expose services via the Internet. The SOA promotes the composition of coarse-grained Web services to build more complex Web applications using standards such as WS-BPEL [20]. Because of the convenience, low cost [1] and capacity to be composed into high-level business processes, Web service technology is becoming increasing popular.

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 [29], 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 [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 [24], since network latency is related to network topology as well as physical distance [11]. To reduce the network latency, WSPs need to allocate Web services to a user concentrated area so that the overall network latency is minimized.

According to a popular Web traffic analyzer, Alexa, 96% of top 1 million Web services were hosted in heterogeneous server clusters or co-location centers [9] that were widely distributed across different network providers and geographic regions. Hence, it is not only necessary but also urgent to provide an effective Web services allocation guide to WSPs so that they can be benefited.

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 main goal of Web service location-allocation is providing a Web service allocation plan to WSPs so that it minimizes the cost as well as optimizes quality of services.

1.1 Objectives

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

1. To model the Web service location-allocation problem so that it can be tackled by Evolutionary Multi-objective Optimization (EMO).

2. To develop a Multi-objective PSO based approach to the Web service location-allocation problem.

1.2 Achievements to Date

In the first trimester, we have accomplished the two objectives. We modelled the service location-allocation problem with a set of matrices as input and a matrix as output. Based on this model, we further developed a multi-objective Particle Swarm Optimization (PSO) based algorithm to solve this problem.

We have conducted a series of experiement that runs on four test cases. The experimental results were fully analyzed by compared between different parameter settings and compared with a NSGA-II based approach. The experiments show that the PSO-based algorithm performed significantly better than NSGA-II based approach in terms of both effectiveness and efficiency. The experiments also show a critical parameter: transfermation threshold, plays an important role in determining the solutions.

1.3 Future Work

We have done the first objective of the project. The second objective is not fully finished for the following reasons: 1.The impact of parameter settings was not fully studied. 2. The effectiveness was not fully verified.

Therefore, in the second half of the project, we mainly focus on: Further improved the algorithm based on analyze the parameter settings. Develop a single-objective based PSO and compare the result of it with the result of multi-objective based PSO.

Chapter 2

Background Survey

2.1 Single-objective methods

Very few researches have studied the Web service location-allocation problem and most of the researchers treat this problem as a single objective problem. [1] [24] try to solve the problem by using integer linear programming techniques. In particular, [24] 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. GA [18] is an Evolutionary algorithm (EA) that uses genetic operators to obtain optimal solutions without any assumptions about the search space. 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.

2.2 Multi-objective methods

So, far, to the best of our knowledge, there is no other researcher has considered the Web service location-allocation problem as a multi-objective problem. In previous research, we proposed a NSGA-II [6] based approach to Web service location-allocation problem. NSGA-II is a multi-objective algorithm based on GA. It is one of the most widely used methods for generating the Pareto frontier, because it can keep diversity without specifying any additional parameters [7]. When it is used for problems with only two objectives, NSGA-II performs relatively well in both convergence and computing speed. However, NSGA-II has been criticized for its high computational cost and bad performance on applications with more than two objectives [8].

NSGA-II 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 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.

Several multiobjective optimization algorithms are based on Particle Swarm Optimization such as Multi-objective Particle Swarm Optimization (MOPSO) [3], Nondominated Sorting Particle Swarm Optimization (NSPSO) [17]. The performance of different multiobjective algorithms was compared in [3] using five test functions. These algorithms are NSGA-II, PAES [16], Micro-GA [4] and MOPSO. The results show that MOPSO was able to generate the best set of nondominated solutions close to the true Pareto front in all test functions except in one function where NSGA-II is superior.

Raquel and et al. [23] proposed a multi-objective Particle Swarm Optimization with crowding distance (MOPSOCD) that extends the MOPSO. The mechanism of crowding distance is incorporated into the algorithm specifically on global best selection and in the delection method of an external archive of nondominated solutions. The diversity of nondominated solutions in the external archive is amintained by using the mechanism of crowding distance together with a mutation operator. The performance shows that MOPSOCD is highly competitive in converging towards the Pareto front and has generated a well-distributed set of nondominated solutions. Another major advantage for PSO-based algorithms is low computation cost. It is much effective than other EC algorithms.

2.3 Summary

Previous researchers have studied Web services location-allocation problem with single-objective algorithms. These approaches have many disadvantages in comparison with multi-objective algorithms. In particular, Evolutionary Multi-objective Optimization algorithms such as NSGA-II and Multi-objective PSO based algorithms show advantages in terms of quality of solution and computational time.

Chapter 3

Work Done

This chapter presents a comprehensive description of the current status of the project. To complete objective one, we designed a matrix-based representation for the service location-allocation problem. A Multi-objective PSO-based approach is then proposed to solve the problem. The experimental results reveal the effectiveness and efficiency of our approach in comparison with our previous NSGA-II-based approach.

3.1 Problem Description , Assumptions and Modeling

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

3.1.1 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.

3.1.2 Assumptions

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

Stakeholder Web Service Providers, User Centers and Candidate Locations

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 first analyze the data collected from current use of services. The collected data should include the records of Web requests from each IP address. Therefore, based on these data, the WSP could summarize 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. Candidate locations are selected based on other criterions such as facilities or deployment cost. User centers and candidate locations can be overlapping. In fact, Web serivce users receive best QoS services if the Web serivces are deployed locally. Therefore, the WSPs would like to choose user centers as candidate locations. In addition to deciding which locations to deploy, information about network latency between user centers and candidate locations are needed.

The list below shows some critical information that should be providered 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 latency from user centers to candidate locations
5. Web service deployment cost for each candidate location

Worth noting that service invocation frequencies are changing over time. For example, a service was popular in some regions may be unfrequented after a few months. That's the main reason for WSPs re-allocate their services. 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.

Static deployment vs. Dynamic deployment

As virtual machine technology and infrastracture-as-a-service (IaaS) are becoming more and more popular. Dynamic Web service deployment become possible [15]. On the other hand, static deployment is still the mainstream because of a majority of Web serivce are deployed on local infrastracture [9]. In this paper, we made an assumption that WSPs periodicaly change the Web service deployment since the user centers are changing over time.

3.1.3 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 a 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-allocation 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.

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

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 [27] [28].

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.

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

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

$$r_{is} = MIN\{l_{ij} \mid j \in \{1, 2, ..., z\} \text{ and } a_{sj} = 1\} \tag{3.1}$$

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:

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

3.2 Multi-objective Particle Swarm Optimization with Crowding Distance for Web Service Location Allocation

To apply MOPSOCD to the service location-allocation problem, the first step is to define variables in MOPSOCD (i.e., to identify particle and the fitness functions).

3.2.1 Particle Representation and Transformation Function

EC algorithms generally treat output or desired solution (e.g., service location-allocation matrix A) as the "individual" so that the solution evolve along with the process. However, in Web service location-allocation, the output is binary. It is not compatible with PSO. In PSO, particles are "flies" in its own direction and velocity searching for a good solution in a continuous space. Therefore, the representation of particle is continuous.

We introduced a *service location-allocation probability matrix*, $A' = [a'_{sj}]$ represents the probability of a service s_i allocate to a candidate location j_i. a'_{sj} is a real value, $a'_{sj} \in (0, 1)$ indicate the probability of a service is **NOT** allocate to a candidate location.

We use the service location-allocation probability matrix A' = $[a'_{sj}]$ as a particle. During the PSO process, the particle needs to be transfered to binary representation in order to compatible with the modeling. In order to transfer $A' \to A$, we introduced a transformation function.

$$a_{sj} = \begin{cases} 1 & \text{if } a'_{sj} > threshold \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

The parameter *threshold* is an empirical parameter that introduced into the algorithm.

3.2.2 Constraint

In our case, we set one basic constraint. The service number constraint requires that each service is deployed in at least one location.

$$\sum_{x \in S} a_{xj} \geq 1 \tag{3.3}$$

3.2.3 Fitness Function

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

$$CostFitness = \sum_{s \in S} \sum_{j \in J} c_{sj} \times a_{sj} \tag{3.4}$$

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

We use *LatencyFitness* to calculate the overall network latency. Where r_{is} denotes the optima response time from a user center i to a service s and f_{is} is the invocation frequency of a user center i to a service s.

$$LatencyFitness = \sum_{i \in I} \sum_{s \in S} r_{is} \times f_{is} \tag{3.5}$$

For example, we use the above mentioned matrices F and R.

$LatencyFitness = f_{11} * r_{11} + f_{12} * r_{12} + f_{13} * r_{13} + ... + f_{33} * r_{33}$
$= 120 * 5.776 + 6.984 * 35 + 0 * 56 + ... + 0.984 * 74$
$= 1300.696$

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 is allocated to all the locations.

$$CostFitness' = \frac{CostFitness - Cost_{min}}{Cost_{max} - Cost_{min}} \tag{3.6}$$

$$LatencyFitness' = \frac{LatencyFitness - Latency_{min}}{Latency_{max} - Latency_{min}} \tag{3.7}$$

3.2.4 MOPSOCD based algorithm for service location-allocation

In this section we present our MOPSOCD based algorithm for service location-allocation as Algorithm 1.

Algorithm 1 MOPSOCD for service location-allocation

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

Outputs: Pareto Front: the *Archive* set

```
Initialize a population P with random real values ∈ (0, 1)
Each individual i in P using the fitness function
Initialize personal best of each individual i.
Initialize GBEST
Initialize Archive with nondominated vectors found in P
repeat
    Compute the crowding distance values of each nondominated solution in the Archive
    Sort the nondominated solutions in Archive in descending crowding distance values
    for ( do each particle)
        Update the GBEST by randomly select a particle from top 10% of P
        Compute the new velocity
        Update its position
        If it goes beyond the boundaries, then multiply its velocity by -1
        If(t < (MAXT * PMUT)), apply Mutation
        Evaluate fitness
        Update its PBESTS
    end for
    Insert new nondominated solution into Archive, remove dominated solutions from Archive
until  maximum iterations is researched
return Archive
```

3.3 Experimental Studies

3.3.1 Dataset

Latency matrix was derived from WS-DREAM [27, 28], 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.

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

3.3.2 Environment

The algorithm was coded in R [19] using existed packages: NSGA2R, MOPSOCD. The program was run on a 3.40GHz desktop computer with 8 GB RAM.

3.3.3 Test case

Four different service location-allocation problems were designed with different complexities.

Table 3.1: Test Cases

problem	number of service	number of candidate location	number of user center
1	20	5	10
2	50	15	20
3	100	25	40
4	200	40	80

3.3.4 Parameter

Parameter settings for MOPSOCD are as follow. The population size is 50 and the maximum number of generations is 50. The mutation rate P_m is 0.5. The inertia parameter w is 0.4. $c1$ and $c2$ are set to 1. The archive size is 250. The transformation threshold is set to 0.7.

Parameter setting for NSGA-II are, population size is 50 and the maximum number of generations is 50. The tournament size is 3. The cross probability P_c is 0.8 and the mutation probability P_m is 0.2.

3.3.5 Evaluation metrics

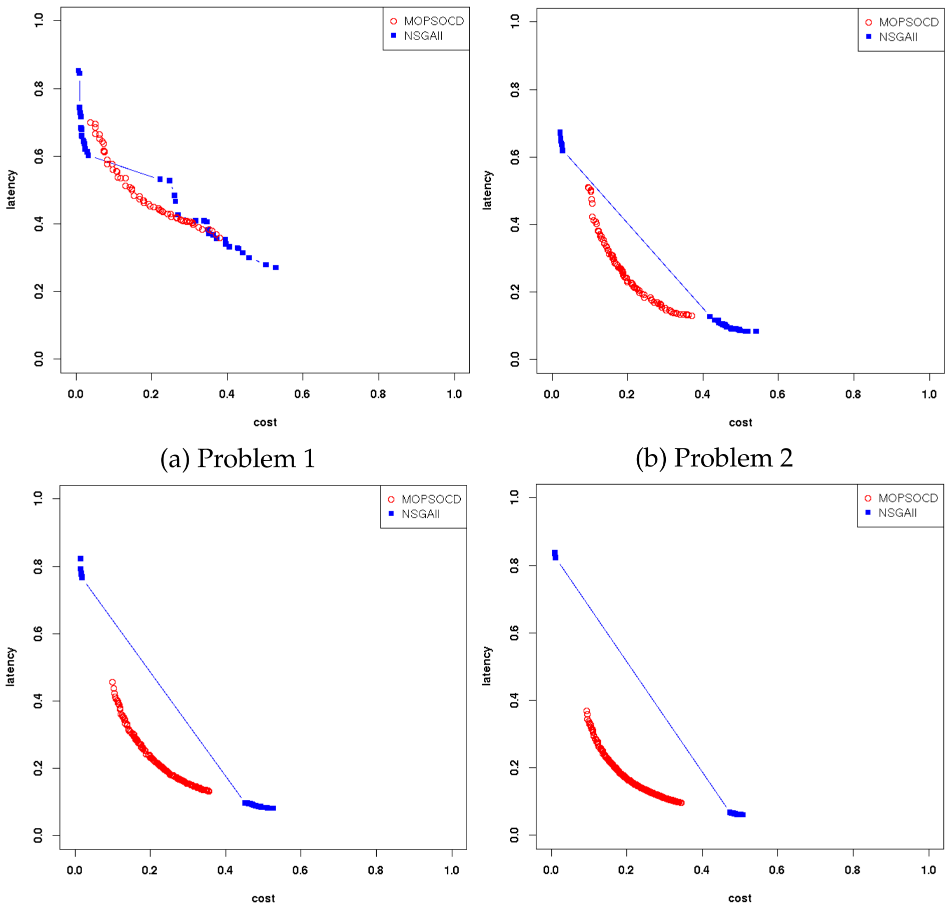
To compare the result of MOPSOCD and NSGA-II, we first derive the Pareto front by using the approach in [25, 26], and then compare the results using approach in [12]. In Xue's approach, 40 sets of solution achieved by each multi-objective algorithm are firstly combined into one set. Secondly, we apply nondominated sort on each solution set and generated final solution set. Thirdly, we use cost fitness value and latency fitness value as x, y coordinate, plot the final solutions on a graph. Our goal is to minimize both cost and latency. Therefore, better solution should locate closer to the origin.

3.3.6 Experimental Results

This section presents an analysis for the proposed MOPSOCD-based approach in solving the service location-allocation problem. Figure 3.1 shows the experimental result for the four test cases.

It is easy to notice that in the first two problems, the number of variable are relatively small. Both algorithms were able to handle the problem and generated a Pareto front. In both test cases, MOPSOCD presents better results. In terms of quality, the Pareto front generated by MOPSOCD is clearly under the NSGA-II which means its closed to optimal solution. In terms of distribution, Pareto front of MOPSOCD is well-distributed along with the true Pareto front. Although in some cases, it sacrificed quality (e.g., upper part of problem 1). On the other hand, NSGA-II form a nonuniform Pareto front.

In the last two problems, the number of variable are huge. Both algorithms show limitations on such scale of variables. As the results show, NSGA-II try to keep diversity in



(a) Problem 1 (b) Problem 2

(c) Problem 3 (d) Problem 4

Figure 3.1: Experimental results

the solution set. As a consequence, the solution set shows polarity among solutions. This is certainly an undesirable solution. In constrast, the result from MOPSOCD shows relatively good coverage of optima Pareto front.

The efficiency of the two algorithms with the results shown in Table 3.2.

Table 3.2: Execution Time (s)

problem 1		problem 2		problem 3		problem 4	
MOPSOCD(s)	NSGA-II(s)	MOPSOCD(s)	NSGA-II(s)	MOPSOCD(s)	NSGA-II(s)	MOPSOCD(s)	NSGA-II(s)
$20.6347 \pm 0.27 \uparrow$	32.79 ± 0.59	$110.26 \pm 0.70 \uparrow$	323.49 ± 9.13	$533.72 \pm 8.63 \uparrow$	2064.127 ± 69.65	$3416.83 \pm 244.03 \uparrow$	18980.26 ± 801.454

As the result shows, MOPSOCD is clearly much efficient than NSGA-II in every test case. Specifically, NSGA-II roughly takes 4 times longer than MOPSOCD in problem 3 and problem 4.

Chapter 4

Conclusion and Future work

4.1 Conclusion

In the project we first developed a model for serivce location-allocation problem so that it can be tackled by evolutionary multi-objective optimization (EMO). Secondly, we developed a multi-objective PSO-based algorithm to solve the problem. The experiments show that our algorithm is both effecient and effective than NSGA-II based algorithm. One of challenge is to find a proper for the value of the transformation threshold. There is no systematic approach to determine the optimal transformation threshold. In order to solve this problem, we mainly consider a binary-based PSO (BPSO) approach. It is natural to apply BPSO on a binary-based representation, therefore BPSO and NSGA-II could use the same representation. In the future, we will further investigate the impact of the parameter settings of multi-objective PSO and developed a single-objective binary PSO to compare with multi-objective based PSO.

A Gantt chart showing the brief plan for the rest of the project is illustrated as follows.

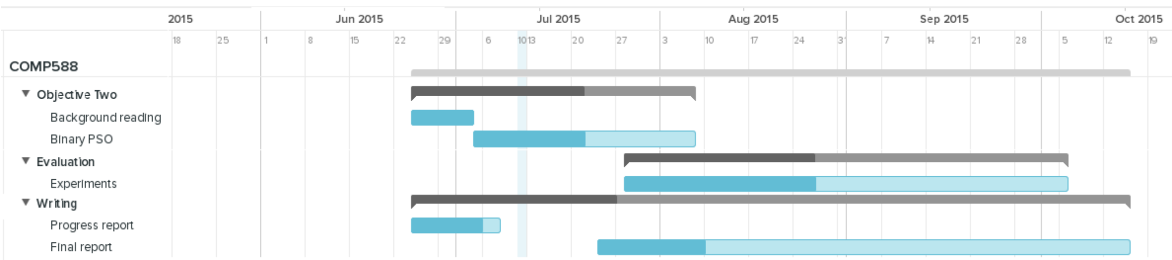


Figure 4.1: A Gantt chart for the rest of the project

Bibliography

[1] ABOOLIAN, R., SUN, Y., AND KOEHLER, G. J. A locationallocation problem for a web services provider in a competitive market. *European Journal of Operational Research 194*, 1 (2009), 64 – 77.

[2] CARAMIA, M. Multi-objective optimization. In *Multi-objective Management in Freight Logistics*. Springer London, 2008, pp. 11–36.

[3] COELLO, C., PULIDO, G., AND LECHUGA, M. Handling multiple objectives with particle swarm optimization. *Evolutionary Computation, IEEE Transactions on 8*, 3 (June 2004), 256–279.

[4] COELLO COELLO COELLO, C., AND TOSCANO PULIDO, G. A micro-genetic algorithm for multiobjective optimization. In *Evolutionary Multi-Criterion Optimization*, E. Zitzler, L. Thiele, K. Deb, C. Coello Coello, and D. Corne, Eds., vol. 1993 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 126–140.

[5] DAN, A., JOHNSON, R. D., AND CARRATO, T. Soa service reuse by design. In *Proceedings of the 2Nd International Workshop on Systems Development in SOA Environments* (New York, NY, USA, 2008), SDSOA '08, ACM, pp. 25–28.

[6] DEB, K., PRATAP, A., AGARWAL, S., AND MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on 6*, 2 (2002), 182–197.

[7] DEB, K., SUNDAR, J., N, U. B. R., AND CHAUDHURI, S. Reference point based multiobjective optimization using evolutionary algorithms. In *International Journal of Computational Intelligence Research* (2006), Springer-Verlag, pp. 635–642.

[8] DOMNGUEZ, J., MONTIEL-ROSS, O., AND SEPLVEDA, R. High-performance architecture for the modified nsga-ii. In *Soft Computing Applications in Optimization, Control, and Recognition*, P. Melin and O. Castillo, Eds., vol. 294 of *Studies in Fuzziness and Soft Computing*. Springer Berlin Heidelberg, 2013, pp. 321–341.

[9] HE, K., FISHER, A., WANG, L., GEMBER, A., AKELLA, A., AND 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* (New York, NY, USA, 2013), IMC '13, ACM, pp. 177–190.

[10] HUANG, H., MA, H., AND ZHANG, M. An enhanced genetic algorithm for web service location-allocation. In *Database and Expert Systems Applications*, H. Decker, L. Lhotsk, S. Link, M. Spies, and R. Wagner, Eds., vol. 8645 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 223–230.

[11] HUFFAKER, B., FOMENKOV, M., PLUMMER, D., MOORE, D., AND CLAFFY, K. Distance Metrics in the Internet. In *IEEE International Telecommunications Symposium (ITS)* (Brazil, Sep 2002), IEEE, pp. 200–202.

[12] ISHIBUCHI, H., NOJIMA, Y., AND 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* (2006), pp. 1143–1150.

[13] JAMIN, S., JIN, C., KURC, A., RAZ, D., AND SHAVITT, Y. Constrained mirror placement on the internet. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (2001), vol. 1, pp. 31–40 vol.1.

[14] JOHANSSON, J. M. On the impact of network latency on distributed systems design. *Inf. Technol. and Management 1*, 3 (2000), 183–194.

[15] KEMPS-SNIJDERS, M., BROUWER, M., KUNST, J. P., AND VISSER, T. Dynamic web service deployment in a cloud environment.

[16] KNOWLES, J., AND CORNE, D. The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on* (1999), vol. 1, pp. –105 Vol. 1.

[17] LI, X. A non-dominated sorting particle swarm optimizer for multiobjective optimization. In *Genetic and Evolutionary Computation GECCO 2003*, E. Cant-Paz, J. Foster, K. Deb, L. Davis, R. Roy, U.-M. OReilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. Potter, A. Schultz, K. Dowsland, N. Jonoska, and J. Miller, Eds., vol. 2723 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003, pp. 37–48.

[18] MAN, K.-F., TANG, K.-S., AND KWONG, S. Genetic algorithms: concepts and applications. *IEEE Transactions on Industrial Electronics 43*, 5 (1996), 519–534.

[19] MORANDAT, F., HILL, B., OSVALD, L., AND 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* (2012), ECOOP'12, Springer-Verlag, pp. 104–131.

[20] ORGANIZATION FOR THE ADVANCEMENT OF STRUCTURED INFORMATION STANDARDS (OASIS). *Web Services Business Process Execution Language (WS-BPEL) Version 2.0*, Apr. 2007.

[21] PAPAZOGLOU, M. P., AND HEUVEL, W.-J. Service oriented architectures: Approaches, technologies and research issues. *The VLDB Journal 16*, 3 (July 2007), 389–415.

[22] RAN, S. A model for web services discovery with QoS. *SIGecom Exch. 4*, 1 (2003), 1–10.

[23] RAQUEL, C. R., AND NAVAL, JR., P. C. An effective use of crowding distance in multiobjective particle swarm optimization. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA, 2005), GECCO '05, ACM, pp. 257–264.

[24] SUN, Y., AND KOEHLER, G. J. A location model for a web service intermediary. *Decis. Support Syst. 42*, 1 (2006), 221–236.

[25] XUE, B., ZHANG, M., AND BROWNE, W. Particle swarm optimization for feature selection in classification: A multi-objective approach. *Cybernetics, IEEE Transactions on 43*, 6 (Dec 2013), 1656–1671.

[26] XUE, B., ZHANG, M., AND BROWNE, W. N. Multi-objective particle swarm optimisation (pso) for feature selection. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation* (2012), GECCO '12, ACM, pp. 81–88.

[27] ZHANG, Y., ZHENG, Z., AND LYU, M. Exploring latent features for memory-based QoS prediction in cloud computing. In *Reliable Distributed Systems (SRDS), 2011 30th IEEE Symposium on* (2011), pp. 1–10.

[28] ZHENG, Z., ZHANG, Y., AND LYU, M. Distributed QoS evaluation for real-world web services. In *Web Services (ICWS), 2010 IEEE International Conference on* (2010), pp. 83–90.

[29] ZHOU, J., AND 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* (2006), pp. 553–557.