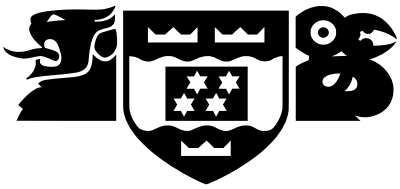
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui



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

The Title

The Author

Supervisor: NOT STATED

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

Abstract

A short description of the project goes here.

Acknowledgments

Any acknowledgments should go in here, between the title page and the table of contents. The acknowledgments do not form a proper chapter, and so don't get a number or appear in the table of contents.

Contents

Figures

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 [19].

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) [3]. 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 [20].

A Web service is a software system allowing to expose services via Internet. The SOA promotes the composition of coarse-grained web services to build more complex web applications using standards such as WS-BPEL [18]. 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 [25], 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 [13]. Study [12] 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 [10]. 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 [8], 96% of Alexa's top 1 million web services were hosted in heterogeneous server clusters or co-location centers that were widely distributed across different network providers and geographic regions. Hence, it is not only necessary but urgent to provide an effective web services allocation guide to WSPs so that they can be benefited.

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 remains high quality of services.

1.1 The Problem

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].

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, integer programming (IP) is very effective for small-scale or mid-scale MKP but suffers from large memory requirement for large-scale MKP [11].

So far, to the best of our knowledge, there is few research has considered using Evolutionary Computation (EC) techniques to solve the location-allocation problem.

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 [14]. The first major challenge is to model the web service location-allocation into a suitable representation so that the EC algorithms and EMO techniques can be employed. The second challenge is develped a binary PSO based and a multi-objective PSO based algorithm to solve this problem.

1.2 Objectives

The aim of this project is to propose two models for service location-allocation problem so that it can be tackled by single-objective and multi-objective evolutionary computation algorithms. Furthermore, we develped a binary PSO based and a multi-objective based approaches to solve the problem. The multi-objective PSO 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 can be divided into three categories:

- Developed two model for the web service location-allocation problem so that it can be tackled by binary PSO and multi-objective PSO.
- To develop a binary PSO and a multi-objective PSO based approach to the web service location-allocation problem.
- To evaluate our proposed approach using some existing datasets.

1.3 Organisation

In Section **??** we introduce the background of multi-objective PSO and NSGA-II. In Section **??** we provide models of the service location allocation problems. Section **??** develops a MOPSOCD based algorithm. The experimental design and results evaluation are shown in Section **??**. Section **??** provides a brief summary.

Chapter 2

Background

2.1 Related work on Service Location-allocation

2.1.1 Traditional Service Location-allocation Methods

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.

2.1.2 EC Approaches(Non-PSO) for Service Location-allocation

Huang [9] 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 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.

In previous research, we proposed a NSGA-II [4] based approach to web service location-allocation problem.

NSGA-II is a multi-objective algorithm based on GA [17]. It is one of the most widely used methods for generating the Pareto frontier, because it can keep diversity without specifying any additional parameters [5]. When 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 [7].

2.2 Evolutionary Computation

2.2.1 Particle Swarm Optimization (PSO)

PSO proposed by Kennedy and Eberhart in 1995 [15]. Similar to evolutionary algorithms (EA), PSO is a population based technique inspired by the swarm behavior of birds and fishes during their search for food. PSO is a meta heuristic in which each individual, called a particle, flies in its own direction and velocity searching for a good solution in the search space. The underlying phenomenon of PSO is optimized by social interaction in the popu-

lation where all particles communicate their information to direct their movement.

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{2.1}$$

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 * r_{1i} * (p_{id} - x_{id}^{t}) + c_2 * r_{2i} * (p_{pg} - x_{id}^{i}) \tag{2.2}$$

PSO is based on the principle that each solution can be represented as a particle in the swarm. Each particle has a random initial position in the search space which is represented by a vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, where D is the dimensionality of the search space. Particles move in the search space to search for the optimal solutions. Each particle has a velocity, represented as $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ which is limited by a predefined maximum velocity, v_{max} and $v_{id} \in$[-v$_{max}, v_{max}$]. During the search process, each particle maintains a record of position its previous best performance, called pbest. The best position of its neighbours is also recorded, which is gbest. The position and velocity of each particle are updated according to the following equations:

In the two equations, t shows the t^th iteration. $d \in D$ shows the d^{th} dimension. w is the inertia weight used to balance the local search and global search abilities of PSO. c_1 and c_2 are acceleration constants. r_{1i} and r_{2i} are random constants uniformly distributed in $[0, 1]$. p_{id} and p_{gd} denote the values of pbest and gbest in d^{th} dimension.

2.2.2 Binary PSO

PSO was originally developed to address continuous optimisation problems. Therefore, the representation for both position and velocity of a particle in PSO is a vector of real numbers. However, this representation is not suitable for many discrete optimisation problems. To address the discrete problem, in 1997 Kennedy and Eberhart developed a binary particle swarm optimisation (BPSO) [16]. In BPSO, the position of each particle is a vector of binary numbers, which are restricted to 1 or 0. The velocity in BPSO represents the probability of the corresponding position taking value of 1. To transfer the velocity value between 0 and 1, we used a sigmoid function to achieve this. The following equation is used to update the position of each particle:

$$x_{id} = \begin{cases} 1 & \text{if } rand() < s(v_{id}) \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

$$s(v_{id}) = \frac{1}{1 + e^{-v_{id}}} \tag{2.4}$$

The rand() is a random number selected from a uniform distribution in $(0, 1)$.

2.2.3 Summary

Chapter 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 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 [23] [24].

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.

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

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 $R = [r_{is}]$,

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

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:

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

Chapter 4

Some LaTeX hints and tips

LaTeX is a very good tool for producing well-structured documents carefully. It is very bad tool for banging things together in a rush and panic.

4.1 Floats

One perennial problem with LaTeX is its treatment of *floats*. Suppose you have a figure or table which you want to include in your document. Where should it go? Traditional typesetting practice is to put these in some convenient place, such as the top or bottom of the current or next page, or at the end of the section or chapter. LaTeX adopts a similar strategy, and allows floats to "float" away from where they were defined. You can give a hint about where you want the figure, but LaTeX may move it. Sometimes this is fine but sometimes you may want to have more control and insist that a float goes *here*. Anselm Lingau's float package gives you this flexibility. For example, the following figure is an example of a non-floating float:

Figure 4.1 The transition function of an NFA with Λ transitions

δ	a	b	Λ
S_1	$\{\}$	$\{\}$	$\{S_2, S_5, S_{10}\}$
S_2	$\{S_3\}$	$\{\}$	$\{\}$
S_3	$\{S_4\}$	$\{\}$	$\{\}$
S_4	$\{S_3\}$	$\{\}$	$\{\}$
S_5	$\{\}$	$\{S_6\}$	$\{\}$
S_6	$\{\}$	$\{S_7\}$	$\{S_8\}$
S_7	$\{S_6\}$	$\{\}$	$\{\}$
S_8	$\{S_9\}$	$\{\}$	$\{\}$
S_9	$\{\}$	$\{S_8\}$	$\{\}$
S_{10}	$\{S_{11}\}$	$\{\}$	$\{\}$
S_{11}	$\{\}$	$\{S_{10}\}$	$\{\}$

On the other hand, Figure 4.2 is a floating float.

You can define different types of new floats, and you can have tables of them in the contents pages.

Figure 4.2 The transition function of an FA to accept the same language.

δ''	a	b
T_1	T_2	T_3
T_2	T_4	T_5
T_3	T_6	T_7
T_4	T_8	
T_5	T_{10}	
T_6		T_{11}
T_7	T_3	
T_8	T_4	
T_{10}		T_5
T_{11}	T_6	

4.2 URL's

Use `\url` from the url package to typeset URL's. Just using `\texttt` or `\tt` does not work:

- `\texttt{http://www.mcs.vuw.ac.nz/~neil/}`
- `\url{http://www.mcs.vuw.ac.nz/~neil/}`

Give:

- `http://www.mcs.vuw.ac.nz/ neil/`
- `http://www.mcs.vuw.ac.nz/~neil/`

If you use the hyperref package then you can produce PDF files with clickable hyperlinks using `\url`.

4.3 Graphics and LaTeX

LaTeX offers rather poor support for the inclusion of graphics. There are lots of ways to include pictorial material in LaTeX, all of which are deficient in some way or other. Look at [**?**] for a description of them. If your document does need to have pictures in it it is worth thinking about what is needed *before* you generate the pictures.

4.4 The bibliography

You should build up your bibliography as you go along. Trying to get the details of the bibliography correct at the end of the project is hard work. Make sure that you record all the relevant details. Beware that material on the internet is likely to change very rapidly. If you are going to include material which is only available on the internet, then you should probably include in the reference the date on which you obtained the document.

4.5 Run LaTeX, run

LaTeX builds up information about your document for the table of contents, references and so on at each run. This means that, for example, the table of contents is really the table of

contents of the previous compilation. You may need to run LaTeX two or three times to let it catch up with itself. If you have cross references within your bibliography (for example two papers from the same collection, such as [**?**, **?**]) you may need to run BibTeX more than once.

It is also possible that the table of contents file has garbage in it, and will prevent the document from being compiled. This may happen if you have had to abort compilation, due to a bug in the source file. If this is the case then removing the `.toc` file will usually solve the problem. You will have to fix the original bug, of course.

4.6 Find out more by...

You can find out more by:

- reading any one of a number of books, such as [**?**, **?**]. The VUW library has copies of these;
- visiting the Comprehensive TeX Archive Network (CTAN) at `www.ctan.org`;
- typing `latex` into Google.

It is *highly unlikely* that you are the first person who ever wanted to do what you want to do with LaTeX. Therefore it is likely that someone has already solved your problem: the real key to using LaTeX well is to make effective use of what other people have done.

4.7 Summary

In this chapter we explained some things about LaTeX.

Chapter 5

Conclusions

The conclusions are presented in this Chapter.

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] 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.

[4] 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.

[5] 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.

[6] DESAI, S., BAHADURE, S., KAZI, F., AND SINGH, N. Article: Multi-objective constrained optimization using discrete mechanics and nsga-ii approach. *International Journal of Computer Applications 57*, 20 (2012), 14–20. Full text available.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] HWANG, J., PARK, S., AND KONG, I. Y. An integer programming-based local search for large-scale maximal covering problems. *International Journal on Computer Science and Engineering* (2011), 837–843.

[12] 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.

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

[14] KANAGARAJAN, D., KARTHIKEYAN, R., PALANIKUMAR, K., AND 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 (2008), 1124–1132.

[15] KENNEDY, J., AND EBERHART, R. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on* (Nov 1995), vol. 4, pp. 1942–1948 vol.4.

[16] KENNEDY, J., AND EBERHART, R. A discrete binary version of the particle swarm algorithm. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on* (Oct 1997), vol. 5, pp. 4104–4108 vol.5.

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

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

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

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

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

[22] VANROMPAY, Y., RIGOLE, P., AND BERBERS, Y. Genetic algorithm-based optimization of service composition and deployment. In *Proceedings of the 3rd International Workshop on Services Integration in Pervasive Environments* (2008), SIPE '08, ACM, pp. 13–18.

[23] 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.

[24] 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.

[25] 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.