A Parallel Cooperative Co-evolutionary Genetic Algorithm for the Composite SaaS Placement Problem in Cloud Computing

Maolin Tang and Zeratul Izzah Mohd Yusoh

Queensland University of Technology,
2 George Street, Brisbane, QLD 4001, Australia

Abstract. A composite SaaS (Software as a Service) is a software that is comprised of several software components and data components. The composite SaaS placement problem is to determine where each of the components should be deployed in a cloud computing environment such that the performance of the composite SaaS is optimal. From the computational point of view, the composite SaaS placement problem is a large-scale combinatorial optimization problem. Thus, an Iterative Cooperative Co-evolutionary Genetic Algorithm (ICCGA) was proposed. The ICCGA can find reasonable quality of solutions. However, its computation time is noticeably slow. Aiming at improving the computation time, we propose an unsynchronized Parallel Cooperative Co-evolutionary Genetic Algorithm (PCCGA) in this paper. Experimental results have shown that the PCCGA not only has quicker computation time, but also generates better quality of solutions than the ICCGA.

1 Introduction

Cloud computing is a new computing paradigm in which all the resources are provided to users as a service over the Internet [1]. According to Gartner, one of the world's leading information technology research and advisory company, by 2014 the cloud computing services revenue is expected to reach 148.8 billion dollars [2]. Software as a Service (SaaS) is one of the most important configurable computing services in cloud computing [3]. It uses a software distribution model in which software is hosted by a SaaS vendor in the cloud and made it available to users as a service over the Internet. SaaS in cloud computing has three distinct characteristics that differentiate itself from a traditional on-premise software. First, it is sold on demand, typically by pay per use. Second, it is elastic - a user can have as much or as little of the service as they want at any given time. Third, the software that provides the service is fully managed by the SaaS vendor. These features allow users to obtain the same benefits of on-premise software without the associated complexity of installation, management, support, licensing, and high initial cost, and therefore make SaaS in cloud computing so compelling.

A composite SaaS is a kind of SaaS that is developed using component-based software development technologies. It usually consists of several software components and data components, such as databases. The composite SaaS placement

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part II, LNCS 7492, pp. 225–234, 2012.

problem is to place those software components and data components on those compute servers and storage servers, respectively, in the cloud such that the performance of the composite SaaS is optimal. The problem is similar to the task assignment problem and the terminal assignment problems addressed in [7,6]. However, the composite SaaS placement problem is more challenging than the task assignment problem as it has more constraints. For example, a software component can be only placed on a compute server and the compute server must meet the CPU and memory requirements of the software component. Thus, the algorithms for the task assignment and the terminal assignment problem cannot be immediately applied to solve the composite SaaS placement problem.

From the computational point of view, the composite SaaS placement is a large-scale combinatorial optimization problem as a cloud may contain thousands of compute severs and storage servers, and a composite SaaS may have dozens of components. Thus, a Penalty-based Genetic Algorithm (PGA) was initially developed [8]. In order to improve the quality of solutions, an Iterative Cooperative Co-evolutionary Genetic Algorithm (ICCGA) was then developed [9]. Experimental results showed that the ICCGA can produce better solutions than the PGA. However, the computation time of the ICCGA was noticeably slow. In order to improve the computation time, this paper presents a Parallel Cooperative Co-evolutionary Genetic Algorithm (PCCGA). The PCCGA has been implemented and tested. Experimental results have shown that the PCCGA not only has quicker computation time, but also produces better solutions, than the ICCGA. In addition, experimental results have shown that the PCCGA has better scalability than the ICCGA.

The remaining paper is organized as follows. Section 2 formulates the composite SaaS placement problem. Section 3 proposes a PCCGA model and entails the PCCGA. Section 4 evaluates the PCCGA. Finally, section 5 concludes this research work.

2 Problem Formulation

Let $C = \{c_1, c_2, \cdots, c_m\}$ be the entire set of m compute servers and $S = \{s_1, s_2, \cdots, s_n\}$ be the complete set of n storage servers in a cloud computing environment. The compute servers and storage servers are interconnected through a set of communication links E. The servers and the communication links together form a cloud communication network. A cloud communication network can be modeled in a graph $G =< V, E >$, where $V = C \cup S$, and if $< v_i, v_j > \in E$ if and only if there exists a communication link between v_i and v_j and $v_i, v_j \in V$.

A composite SaaS, X, consists of a set of software components $SC = \{sc_1, sc_2, \cdots, sc_p\}$ and a set of data components $SS = \{ss_1, ss_2, \cdots, ss_q\}$, where p is the number of software components and q the number of data components in X. The control dependencies and data dependencies between those software components are stored in sets CD and DD, respectively.

A SaaS component has a CPU requirement and a memory requirement. A compute server has a CPU capacity and a memory capacity. A software component

$sc_i =< sc_i^{cpu}, sc_i^{mem} >$ can be deployed on a compute server $c_j =< c_j^{cpu}, c_j^{mem} >$ only when the compute serve c_j can meet both the CPU and memory requirements of the software component sc_i, that is $sc_i^{cpu} \leq c_j^{cpu}$ and $sc_i^{mem} \leq c_j^{mem}$, where $1 \leq i \leq p$ and $1 \leq j \leq m$.

Similarly, a data component has a space requirement and a storage server has a space capacity. A data component $sd_i =< sd_i^{space} >$ can be placed on a storage server $s_j =< s_j^{space} >$ only when the storage server has enough room to hold the data component sd_i, that is $sd_i^{space} \leq s_j^{space}$, where $1 \leq i \leq q$ and $1 \leq j \leq n$.

Given a composite SaaS $X =< SC, SS, CD, DD >$ and a cloud computing communication network $G =< C \cup S, E >$, the composite SaaS placement problem is to find $f_1 : SC \rightarrow C$ and $f_2 : SS \rightarrow S$ such that the performance of the composite SaaS is optimal measured by the Estimated Execution Time of the composite SaaS, which is derived in [9].

3 Parallel Cooperative Co-evolutionary Genetic Algorithm

This section presents an unsynchronized parallel computation model and describes the algorithm of the PCCGA.

3.1 Parallel Model

The parallel model based on which the PCCGA is developed is derived from a cooperative co-evolution model proposed by Potter and de Jong [10]. In the cooperative co-evolution model, a problem is divided into several interacting subproblems. For each of the subproblems, an evolutionary algorithm, such as genetic algorithm, is used to solve it independently, and the multiple subproblems are solved concurrently using multiple independent evolutionary algorithms. The interaction between the evolutionary algorithms occurs only when evaluating the fitness value of an individual in the population of an evolutionary algorithm as the individual is only part of the solution to the problem in the domain and therefore in order to evaluate its fitness the PCCGA needs to combine the individual with a representative from each of the other evolutionary algorithms to form a complete solution. An individual is rewarded when it works well with the representative from the other evolutionary algorithms and is punished when it does not work well with the representative.

Based on the cooperative co-evolutionary model, we developed an unsynchronized parallel model as shown in Fig. 1. In the parallel model, we decompose the computation into two unsynchronized and parallel sub-computations. One of the sub-computations is the placement of the software components; another the placement of the data components. For each of the sub-computations, we use a genetic algorithm (GA) to solve it. The communication between the two GAs is asynchronous through a buffer. The buffer has two units. One unit keeps the best solution from the software component placement sub-computation; the other the best solution from the the data component placement sub-computation. At the

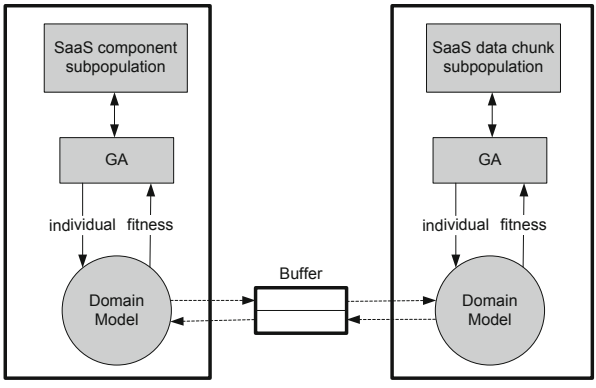

Fig. 1. Parallel Cooperative Co-evolutionary Genetic Algorithm Model

end of each generation, the GAs update their best solution in the buffer. Since the two GAs are not synchronized, one GA may update its best solution in the buffer more frequently than the other during the computation.

3.2 Algorithm Descriptions

The PCCGA invokes a classical GA for the software component placement problem and a classical GA for the data component placement problem. Thus, before giving the algorithm description of the PCCGA, we present the two classical GAs. The encoding scheme and genetic operators used in the GAs are the same with those in [9].

The GA for the Software Component Placement Problem

1. randomly generate an initial population of solutions to the software component placement problem;
2. while the termination condition is not true
   (a) get the best solution to the data component placement problem from the buffer;
   (b) for each individual in the population:
      i. combine the individual with the best solution to the data component placement problem to form a complete SaaS placement solution;
      ii. calculate the fitness value of the SaaS placement solution.
   (c) select individuals for recombination from the population based on their fitness values and pair them up;
   (d) probabilistically apply the crossover operator to each of the pairs to generate new individuals;
   (e) probabilistically use for the mutation operator to each of the new individuals;
   (f) use the new individuals to replace the old individuals in the population;
   (g) update the best software component placement solution in the buffer.

The GA for the Data Component Placement Problem

1. randomly generate an initial population of solutions to the data component placement problem;
2. while the termination condition is not true
   (a) get the best solution to the software component placement problem from the buffer;
   (b) for each individual in the population:
      i. combine the individual with the best solution to the software component placement problem to form a complete SaaS placement solution;
      ii. calculate the fitness value of the SaaS placement solution.
   (c) select individuals for recombination from the population based on their fitness values and pair them up;
   (d) probabilistically apply the crossover operator to each of the pairs to generate new individuals;
   (e) probabilistically use for the mutation operator to each of the new individuals;
   (f) use the new individuals to replace the old individuals in the population;
   (g) update the best data component placement solution in the buffer.

The Algorithm Description of the PCCGA

1. while the termination condition is not true
   (a) run the GA for the software component placement problem and the GA for the data component placement problem in parallel;
2. combine the best solution to the software component placement problem and the best solution to the data component placement problem in the buffer to form a solution to the SaaS placement and output it.

4 Evaluation

This section evaluates the performance of the PCCGA, including its computation time, quality of solution and scalability. Since there is no benchmark available for the composite SaaS placement problem, we have to use the performance of the ICCGA as a benchmark.

In order to conduct a comparative study of the PCCGA and the ICCGA, we implemented both of them in Microsoft Visual Studio C#. We also developed a C# program to randomly generate a cloud communication network of a given configuration based on the cloud model presented in [11] and another C# program to randomly create a composite SaaS placement problem of a given configuration.

Since the complexity of a composite SaaS placement problem depends on both the size of the cloud communication network and the size of the composite SaaS placement problem, we conducted two groups of experiments. In the first group of experiments, we randomly generated a cloud communication network that

Table 1. The characteristics of the five composite SaaS placement problems

Test	Test Problem Characteristics		
Problem	No. of software comp.	No. of data comp.	Total no. of comp.
1	5	5	10
2	10	10	20
3	15	15	30
4	20	20	40
5	25	25	50

has 100 compute servers and 100 storage servers, and then randomly generated five composite SaaS placement problems of different sizes. Table 1 shows the characteristics of the five composite SaaS placement problems.

This group of experiments were designed to evaluate the speed-up ratio and the quality of the solutions of the PCCGA when it is used for solving different sizes of composite SaaS placement problems and to study how the computation time of the PCCGA would increase when the size of the composite SaaS placement problem increases. We used both the PCCGA and the ICCGA to solve each of the five composite SaaS placement problems. Considering the stochastic nature of the PCCGA and the ICCGA, for each of the composite SaaS placement problems we repeated the experiment for 10 times and recorded the quality of the solutions generated by both of the algorithms and their computation times. The statistics about the computation time and the quality of solutions of the two algorithms are shown in Table 2 and Table 3 respectively.

It can be seen from Table 2 that the average computation time of the PCCGA is between 11.29% and 37.15% of the average computation time of the ICCGA for the five test problems. However, the average estimated execution time of the composite SaaS placement produced by the PCCGA is also better than that produced by the ICCGA (the average estimated execution time of the PCCGA is only between 23% and 53% of that of the ICCGA).

In addition, in the evaluation, we also compared the scalability of the PCCGA with the scalability of the ICCGA. Fig. 2 displays how the average computation times increased when the size of the composite SaaS increased. When the total number of SaaS components and data components increased from 10 to 50, the average computation time of the ICCGA increased from 278.5 seconds to 9266.8 seconds, while the computation time of the PCCGA only increased from 103.3 seconds to 1046.9 seconds linearly.

In the second group of experiments, we randomly generated a composite SaaS placement problem that has 10 software components and 10 data components, and randomly generated five cloud communication networks. Table 4 shows the characteristics of the five randomly generated cloud communication networks.

Then, we used both the PCCGA and the ICCGA to solve the composite SaaS placement problem in the five cloud communication networks of different sizes. Considering the stochastic nature of the two algorithms, for each of the experiments we repeated for 10 times and recorded the qualities of the solutions and the computation times for each run of the experiments. The statistics about

Table 2. Comparison of the computation times of the PCCGA and the ICCGA for different composite SaaS placement problems

Test	PCCGA (second)				ICCGA (second)			
Problem	Best	Worst	Ave	SD	Best	Worst	Ave	SD
1	69.6	154.1	103.5	27.4	188.4	419.4	278.5	96.2
2	132.6	371.4	231.5	80.4	1020.0	2416.8	1557.1	415.8
3	382.8	832.2	621.0	144.9	1743.6	6877.8	3859.9	1541.3
4	415.2	950.4	710.3	174.1	2466.0	8473.8	5505.0	2103.3
5	632.4	1938	1046.9	403.6	3436.2	15763.8	9266.82	3755.8

Table 3. Comparison of the qualities of solutions produced by the PCCGA and the ICCGA for different composite SaaS placement problems

Test	PCCGA (millisecond)				ICCGA (millisecond)			
Problem	Best	Worst	Ave	SD	Best	Worst	Ave	SD
1	513	89309	24958.5	29056.3	89748	129609	108866.9	12613.6
2	112	180144	36680.2	53964.5	107524	263752	140893.0	46928.2
3	22042	114729	72857.1	36736.7	130569	235881	172433.3	33817.7
4	12674	180144	129565.7	63641	174319	263752	243215.3	47358.1
5	16100	216704	123456.6	88892.8	217730	648594	316120.1	120410.2

Table 4. The characteristics of the clouds

Test	Test Problem Characteristics		
Problem	No. of compute servers	No. of storage servers	Total no. of servers
1	50	50	100
2	100	100	200
3	150	150	300
4	200	200	400
5	250	250	500

the computation times and the quality of solutions of the two algorithms are shown in Table 5 and Table 6 respectively.

It can be seen from Table 6 that the average computation time of the PCCGA is between 17.66% and 31.89% of the average computation time of the ICCGA for the five test problems. However, the average estimated execution time of the composite SaaS placement produced by the PCCGA is also better than that produced by the ICCGA (the average estimated execution of the PCCGA is only between 45% and 55% of that of the ICCGA).

In addition, in the evaluation, we also compared the scalability of the PCCGA with the scalability of the ICCGA. Fig. 3 displays how the average computation times increased when the size of the composite SaaS increased. When the total number of compute and storage servers in cloud computing increased from 100 to

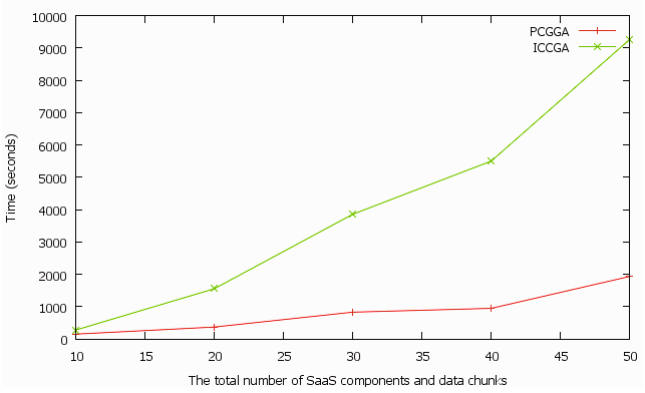

Fig. 2. The computation time increasing trend when the size of the composite SaaS increases

Table 5. Comparison of the computation times of the PCCGA and the ICCGA in different clouds

Test	PCCGA (sec)				ICCGA (sec)			
Problem	Best	Worst	Ave	SD	Best	Worst	Ave	SD
1	54.0	136.2	89.4	31.3	145.2	440.4	280.3	88.5
2	115.8	440.9	288.5	107.2	813.0	2228.4	1362.4	482.3
3	233.4	1302.0	672.4	326.3	712.8	3451.2	2456.0	809.7
4	401.4	1404.6	774.6	316.9	3024.0	8580.0	4385.4	1629.9
5	771.6	1707.6	1350.5	309.9	3885.6	10029.6	7187.6	1824.2

Table 6. Comparison of the qualities of solutions produced by the PCCGA and the ICCGA in different clouds

Test	PCCGA (millisecond)				ICCGA (millisecond)			
Problem	Best	Worst	Ave	SD	Best	Worst	Ave	SD
1	7284	53536	28519.9	15211.9	48064	79026	66425.4	9186.6
2	34357	110918	61238.9	24669.8	97370	168355	137089.4	21842.2
3	34588	254672	162969.4	65807.6	202857	379929	295677.8	47531.4
4	120114	359175	241791.2	77513.7	368345	623340	501810.4	73681.5
5	44786	509168	274609.9	136396.5	363253	623821	547364.0	80760.1

500, the average computation time of the ICCGA increased from 280.3 seconds to 7187.6 seconds, while the computation time of the PCCGA only increased from 89.4 seconds to 1350.5 seconds linearly.

In all the experiments, the subpopulation sizes for the compute server GA and the storage server GA were set at 100 in both the PCCGA ad the ICCGA. The probabilities for crossover and mutation were set at 0.95 and 0.15, respectively,

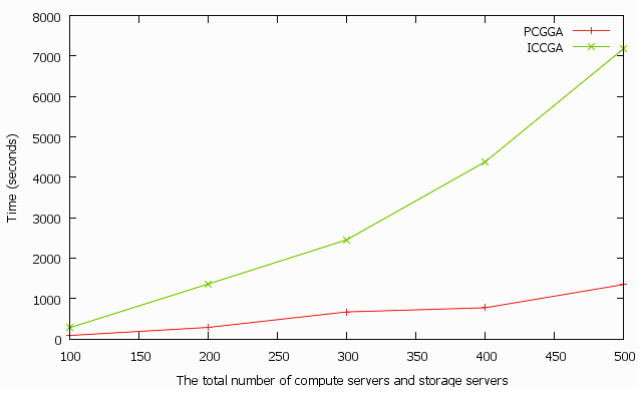

Fig. 3. The computation time increasing trend when the size of cloud computing increases

in both the PCCGA and the ICCGA. The termination condition used in both the PCCGA and the ICCGA was 'no improvement on the best solution for 25 consecutive generations'. All the experiments were carried out in a computer with 3.00 GHz Intel Core 2 Duo CPU and 4GB RAM.

5 Conclusion and Future Work

This paper has proposed an unsynchronized parallel cooperative co-evolutionary genetic algorithm (PCCGA) for the composite SaaS placement problem in cloud computing, and has evaluated the performance of the PCCGA by comparing it with an iterative cooperative co-evolutionary genetic algorithm (ICCGA). The experimental results have shown that the computation time of the PCCGA was noticeably quicker than that of the ICCGA. In addition, the experimental results have shown that on average the quality of the solutions produced by the PCCGA is much better than that of the ICCGA for those randomly generated test problems. Moreover, the experimental results have shown that the PCCGA has better scalability than the ICCGA. To the best of our knowledge, this research is the first attempt to tackle the composite SaaS placement algorithm using parallel evolutionary computation.

The SaaS placement is a large-scale complex combinatorial optimization. Thus, how to further improve its computation time by increasing its parallelism is an issue that we will investigate in the future, and one possible way to improve the parallelism is to break down the composite SaaS placement problem into more subcomponents using the random grouping techniques proposed in [12]. In addition, in the PCCGA we always select the best individual from the other population, which may not be appropriate in some cases. Thus, another work that I will do in the future is to the selection strategy.

Acknowledgment. The programs used in the evaluation were developed by Peter Wong at Queensland University of Technology. In addition, the authors would like to thank the PC members and reviewers of this papers for your valuable comments.

References

1. Foster, I., Yong, Z., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Proceeding of Grid Computing Environment Workshop, pp. 1–10 (2008)
2. Gartner Inc., Gartner says worldwide cloud services market to surpass $68 Billion in 2010 (2010), http://www.gartner.com/it/page.jsp?id=1389313
3. Armbrust, M., et al.: Above the clouds: a Berkeley view of cloud computing. Tech. Rep. UCB/EECS-2009-28, EECS Department, U.C. Berkeley (2009)
4. Khare, V., Yao, X., Sendhoff, B.: Multi-network evolutionary systems and automatic problem decomposition. International Journal of General Systems 35(3), 259–274 (2006)
5. Khare, V.R., Yao, X., Sendhoff, B.: Credit Assignment Among Neurons in Coevolving Populations. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN VIII. LNCS, vol. 3242, pp. 882–891. Springer, Heidelberg (2004)
6. Salcedo-Sanz, S., Xu, Y., Yao, X.: Hybrid meta-heuristics algorithms for task assignment in heterogeneous computing systems. Computers and Operations Research 33(3), 820–835 (2006)
7. Salcedo-Sanz, S., Yao, X.: A hybrid hopfield network – genetic algorithm approach for the terminal assignment problem. IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 34(6), 2343–2353 (2004)
8. Yusoh, Z., Tang, M.: A penalty-based genetic algorithm for the composite SaaS placement problem in the cloud. In: Proceeding of IEEE World Congress on Computational Intelligence, pp. 600–607. IEEE, Spain (2010)
9. Yusoh, Z.I.M., Tang, M.: A Cooperative Coevolutionary Algorithm for the Composite SaaS Placement Problem in the Cloud. In: Wong, K.W., Mendis, B.S.U., Bouzerdoum, A. (eds.) ICONIP 2010, Part I. LNCS, vol. 6443, pp. 618–625. Springer, Heidelberg (2010)
10. Potter, M., de Jong, K.: A Cooperative Coevolutionary approach to Function Optimization. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN III. LNCS, vol. 866, pp. 249–257. Springer, Heidelberg (1994)
11. IBM System Storage, http://www-07.ibm.com/storage/au/
12. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Information Sciences 178(15), 2985–2999 (2008)