A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling

Zhangjun Wu[1,2], Zhiwei Ni[1], Lichuan Gu[1]
[1]Institute of Intelligent Management
Hefei University of Technology
Hefei, China
wuzhangjun@mail.hfut.edu.cn
nzwgd@hfut.edu.cn
gulichuan@ahan.edu.cn

Xiao Liu[2]
[2]Faculty of Information and Communication Technologies
Swinburne University of Technology
Melbourne, Australia
xliu@swin.edu.au

***Abstract*-A cloud workflow system is a type of platform service which facilitates the automation of distributed applications based on the novel cloud infrastructure. Compared with grid environment, data transfer is a big overhead for cloud workflows due to the market-oriented business model in the cloud environments. In this paper, a Revised Discrete Particle Swarm Optimization (RDPSO) is proposed to schedule applications among cloud services that takes both data transmission cost and computation cost into account. Experiment is conducted with a set of workflow applications by varying their data communication costs and computation costs according to a cloud price model. Comparison is made on makespan and cost optimization ratio and the cost savings with RDPSO, the standard PSO and BRS (Best Resource Selection) algorithm. Experimental results show that the proposed RDPSO algorithm can achieve much more cost savings and better performance on makespan and cost optimization.**

Keywords: discrete particle swarm optimization, workflow scheduling, cloud computing.

I. INTRODUCTION

Cloud computing is emerging as the latest distributed computing paradigm and attracts increasing interests of researchers in the area of Distributed and Parallel Computing[1], Service Oriented Computing[2] and Software Engineering[3]. Generally speaking, the function of a cloud workflow system and its role in a cloud computing environment, is to facilitate the automation of user submitted workflow applications where the tasks have precedence relationships defined by graph-based modeling tools such as DAG (directed acyclic graph) and Petri Nets[4], or language-based modeling tools such as XPDL (XML Process Definition Language)[17].

Among many others, one of the most important aspects which differentiate a cloud workflow system from its other counterparts is the market-oriented business model. Such a seemed small change actually brings significant innovations to conventional computing paradigms since they are usually based on non-business community models where resources are shared and free to be accessed by community members[5]. Meanwhile, application data can be hosted on different storage resources at the global cloud infrastructure. When one task needs to process data from different data centers, moving the data becomes a challenge[6]. In order to efficiently and cost effectively schedule the tasks and data of applications among cloud services, end user QoS-based scheduling strategies are implemented, such as those for minimizing makespan, minimizing total execution cost and balancing the load of resources[7]. In this paper, we focus on minimizing the execution time and the execution cost of applications on these resources provided by Cloud service providers, such as Cisco and Amazon.

The particle swarm method for function optimization has been introduced by Kennedy and Eberhart in[8]. The ability of groups of some species of animals to work as a whole in locating desirable positions in a given area is simulated. It has better ability of global searching and has been successfully applied to many areas[9]. This algorithm is predominately employed to find solutions for continuous problem without prior information. Unfortunately, workflow scheduling which is one of a variety of NP-completes is a discrete and very complicated optimization issue. Several approaches have been developed for PSO to solve discrete problem, such as swap operation[10], angle modulation[11], space transformation[12] and priority-based representation [13]. Although various discrete PSO variants have been proposed, their performance is generally not satisfactory when compared with other meta-heuristics for discrete optimization [18].

More recently, set-based concept is introduced into PSO to solve combinatorial optimization problems, such as determining RNA secondary structure[14],traveling salesman problem (TSP) and multidimensional knapsack problem (MKP)[15]. This concept has been proved to be promising. Based on the set-based scheme, we use RDPSO to minimize the total computation cost of cloud workflow.

The rest of this paper is organized as follows. Section 2 briefly describes the scheduling model in workflow, and section 3 describes the proposed algorithm in detail. Section 4 shows the experimental results. Finally, section 5 addresses the conclusions.

II. WORKFLOW TASK LEVEL SCHEDULING

A. Workflow Application Model

A cloud workflow application can commonly be modelled

978-0-7695-4297-3/10 $26.00 © 2010 IEEE
DOI 10.1109/CIS.2010.46

as a Directed Acyclic Graph, denoted as $(\mathrm{DAG}): G=(V, A)$. The set of nodes $V=\{T_1, T_2, ..., T_n\}$ represents the tasks in the workflow application; the set of arcs denotes precedence constraints and the data dependencies between tasks. An arc is in the form of $d_{i,j}=(T_i, T_j) \in A$ where T_i is called the parent task of T_j, T_j is the child task of T_i, $d_{i,j}$ is the data produced by T_i and consumed by T_j. We assume that a child task cannot be executed until all of its parent tasks have been completed.

Suppose that n tasks are to be scheduled on m service instances. For each task $T_i(1 \le i \le n)$ in the workflow, there are a set of candidate service instances $s_i=(s_i^1, s_i^2, ..., s_i^{m_i})$ and a set of storage sites $D=(D_1, D_2, ..., D_{m_i})$ available, where $s_i^j(1 \le m_i)$ represents a service instance provided by a GSP (Global Service Provider) and m_i is the total number of service instances for T_i. The properties of a service instance s_i^j can be represented as a group of three variables $(s_i^j \cdot g, s_i^j \cdot t, s_i^j \cdot c)$, in which $s_i^j \cdot g$ stands for the GSP of s_i^j while $s_i^j \cdot t$ and $s_i^j \cdot c$ denote the execution time and cost of s_i^j, respectively.

Figure 1. A workflow with 9 tasks

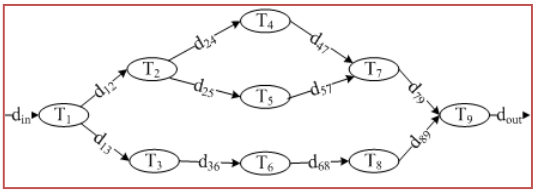

Figure 2. Service instances and data storage

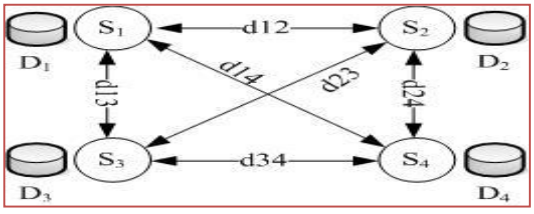

Fig. 1 shows a workflow application with 9 tasks, in which the each task can be implemented by four service instances. Fig. 2 shows the four service instances and the data storage, they are fully connected and symmetric.

B. Task level scheduling Objectives

The task level scheduling mainly consists of the following three steps: 1) Obtain the QoS constraints (in this paper QoS constraints refer to makespan and cost. These are varying to the user's specific quality preference and their budget) for each individual tasks. 2) Optimize the task-service assignments. In cloud data centers, the underlying resources are virtualized and can be created dynamically to suit the needs of different cloud applications. 3) Implement the optimal scheduling plan. Based on the above three steps, a task level scheduling plan is implemented among the cloud data centers to carry out the workflow execution with satisfactory QoS. Meanwhile, the overall running cost of the cloud workflow system has also been minimized.

There are several objectives can be measured for the mapping of workflow tasks to distributed service. In the cloud computing environment, computation cost is usually the first priority of user's concern. In this paper, we focus not only on minimizing the total execution cost but also on minimizing the total makespan of the workflow application. Therefore, the major goal for our task level scheduling is to decrease the computation cost on the condition of satisfying the deadline of cloud workflow application by dynamically optimizing the Task-to-Resource assignment.

Let T_{s_i} be the total makespan of the service S_i, $T_{total}(M)$ be the total makespan (the overall completion time) of the workflow application. It is the maximum of all the services makespan. That is

$$T_{total}(M)=\max(T_{s_i}) \quad (1)$$

Let $C_{exe}(M)$ be the total execution cost of the workflow scheduling M, $C_{trans}(M)$ is the total data transmission cost of the workflow scheduling M, $C_{total}(M)$ be the total computation cost of the workflow scheduling M. The computation cost of a scheduling consists of the execution cost and the data transfer cost.

$$C_{total}(M)=C_{exe}(M)+C_{trans}(M) \quad (2)$$

The price for transferring basic data unit (e.g. per Mb) between two services and the price for computation of basic time unit (e.g. per hour) are given by the service providers. The cost of communication is applicable only when two tasks have data dependency between them. Usually, there is no data transfer charge within the same region of the same service provider. The objective function of this paper can be defined as

$$Minimize(T_{total}(M)+C_{total}(M)) \quad (3)$$

Equation 3 ensures that all the tasks are not mapped to a single service. Relatively heavy cost will be required to initially distribute tasks to all resources. Subsequent minimization of the overall cost ensures that the total cost is minimal even after initial distribution. For a given assignment M, the total cost $C_{total}(M)$ for a service is the sum of execution cost.

III. THE PROPOSED ALGORITHM

A. Particle Swarm Optimizer

The original PSO algorithm was inspired by the social behavior of biological organisms. Suppose that the searching space is *D-dimensional* with *N* randomly initialized particles in it. Each particle is represented by a *D-dimensional* vector X_i *(i=1, 2⋯ d)* which stands for its location $(x_{i1}, x_{i2}⋯, x_{in})$ in space and it is also regarded as a potential solution. The position of the best individual of the whole swarm is noted as the global best position P_g, and the fitness of the global best position is noted as the global best fitness F_g. Then the velocity of particle and its new position will be updated according to the following two equations:

$$v_{id}^{n+1} = \chi(wv_{id}^{n} + c_1 r_1(p_{id}^{n} - x_{id}^{n}) + c_2 r_2(p_{gd}^{n} - x_{id}^{n})) \quad (4)$$

$$x_{id}^{n+1} = x_{id}^{n} + v_{id}^{n} \quad (5)$$

Where, $d=1,2,⋯,D$; $i=1,2,⋯,N$; $n=1,2,⋯,iter_{max}$ ($\text{iter}_{\max}$ is the allowed largest iteration step); w is called inertia weight; c_1 and c_2 are two positive constants called acceleration coefficients; χ is a constriction factor, which is used to limit the maximum velocity; r_1 and r_2 are two random numbers uniformly from the interval [0, 1].

B. The Revised Discrete Swarm Model

The concept of particle swarm is originally designed to find solutions for continuous optimization problems without prior information. To solve the workflow scheduling problem, a revised discrete version of PSO (RDPSO) based on the concept of set-based is adopted in this paper. Similar to conventional PSO, the key issue of DPSO is to define the position and velocity of particle as well as to define their operation rules and the equation of motion according to the features of discrete variables.

For the sake of clarity, variables and the rules of DPSO for solving workflow scheduling can be depicted in definitions. In general, a mapping of workflow can be defined by a set of pairs $M = [< T_i, S_j >](i \in [1,m], j \in [1,n])$. m is the number of tasks to be scheduled and n is the number of services available in the cloud environment.

Definition 1: A *position* is a feasible solution to the scheduling problem and consists of a set of <*task, service*> pairs. Each pair means a mapping that task is mapped onto a service .It also indicate that the position of each particle satisfies the precedence constraint between activities.

Definition 2: A *velocity* is a set with possibilities. $V = \{e / p(e) \mid e \in E\}$, E is the set of <*task, service*> pairs. $p(e) \in [0,1]$, it shows the possibility of the task mapping to the service.

Definition 3: Subtraction between two particle positions, named as x_1 and x_2, is defined as a set of pairs which exist in x_1 but not in x_2.This operator also know as relative complement or set theoretic difference between two sets.

Definition 4: Multiplication between random real number and velocity is defined as $cV = \{e / p'(e) \mid e \in E\}$.

$$p'(e) = \begin{cases} 1, & if \quad c * p(e) > 1 \\ c * p(e), & otherwise \end{cases}$$

Definition5: Addition of two velocities is defined as reservation of the larger one.

$$V_1 + V_2 = \{e / \max(p_1(e), p_2(e)) \mid e \in E\}$$

Definition 6: New position generation is defined as a constructive procedure. The constraints between tasks must be taken into account. The <*task, service*> pairs in the new X_i^j can come from *gbest*, *pbest*, *previous position* or other *feasible pairs*.

RDPSO based task level Scheduling

```
01 swarm initialization with GRASP;
02 calculate pbest and gbest;
03 while(stop criterion is not meted)
04 calculate V_p and V_g ;
05 construct the new position x_{i+1}^j ;
   1: select pairs from P(V_i^j) = {e | e / p(e) ∈ V_i^j};
   2: if x_{i+1}^j is not completed, select pairs from x_i^j ;
   3: else if x_{i+1}^j is not completed, select pairs from
   other feasible pairs;
06 calculate fitness value;
07 update pbest and gbest;
08 end while;
09 return gbest;
```

First, the algorithm starts with swarm initialization using GRASP (greedy randomized adaptive search procedure) to ensure each particle in the initial swarm is a feasible and efficient solution. Then, compute the potential exemplars, pbest and gbest, for particles to learn from while they are moving. The stop condition can be the user's QoS requirements, such as deadline, the budget for computation cost or data transfer cost. The particle's new position generation procedure has three steps: 1) select elements from the promising set of pairs with larger probability, that is, the particle learns from gbest and pbest; 2) due to the discrete property of scheduling, there are usually not enough feasible pairs in gbest to generate new position, so the particle will learn from its previous position; 3) all the unmapped tasks should choose resources from other feasible pairs. Fitness function can be defined according to the objectives mentioned in 2.2. Finally, gbest will be return as optimal solution.

IV. EVLUATION

In this section we describe the experimental settings, algorithm settings, the experimental results and discussions.

A. Environment and algorithm settings

Assume all tasks are executed on the Amazon Elastic Compute Cloud (http://aws.amazon.com), all the data are stored in Amazon Simple Storage Service and data transmissions are fulfilled through the Amazon Cloud Front. And assume that Service 1 and 2 to be in US, Service 3 in Euro and Service 4 in APAC. Due to the varying price of service, in the following simulation, the price at this moment is adopted.

Cost of execution of T_i on $Service_j$ is \$0.17 per hour (resources for high-CPU, on-demand instance medium instances, Linux Usage). *Taskcost = Tasktime * Price*.

Data communication unit cost matrix is shown in Table 1. Each task has own input/output data and the sum of all data in the matrix varies according to the data size we test (64-2048M).

TABLE I. COST MATRIX OF DATA COMMUNICATION (\$/MB/SECOND)

	S_1	S_2	S_3	S_4
S_1	0	0.01	0.15	0.19
S_2	0.01	0	0.15	0.19
S_3	0.15	0.15	0	0.20
S_4	0.19	0.19	0.20	0

TABLE II. DATA INPUT/OUTPUT FOR EACH TASK (MB)

	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
input	10	10	12	12	15	20	18	29
output	12	15	8	12	18	17	12	25

As for workflow, the number of total tasks ranges from 50 to 300 including both workflow and non-workflow activities. The number of workflow segments increase accordingly from 5 to 50. The number of resources is constrained in the range of 3 to 20. QoS constraints including time constraint and cost constraint for each task are defined as follows: time constraint is defined as the mean duration plus 1.28* $\sqrt{variance}$ and cost constraint is defined as the triple of the corresponding time constraint. The makespan of a workflow is defined as the latest finished time on all the virtual machines and the total cost of a workflow is defined as the sum of task durations multiply the prices of their allocated virtual machines. These settings are much similar to the settings in[16].

As for RDPSO, the settings for parameters are: Population size, n=50; the constriction factor $\chi = 0.73$; c1=c2= 2.0; Maximum fitness evaluations, me=3000; Refreshing gap, m=7; Inertia weight, iwt = 0.9 - (1: me) * (0.7/me); Pc=0.05+0.45*(exp (10(i-1)/ (ps-1))-1)/ (exp (10)-1).

B. Results and Analysis

First, we varied the size of total data processed by the workflow depicted in Fig.1 in the range from 64M to 2G. The total costs for different data size are compared. The data in table 3 are the computation cost of the workflow with the increase in the total data processed by the workflow.

The cost obtained by RDPSO or PSO based task-service mapping increases much lower than the BRS algorithm. The main reason for two PSOs to perform better than the BRS is the way that they take communication costs of all the tasks, including dependencies between them into account. However, the BRS algorithm calculates the cost for a single task at a time, which does not consider the mapping of other tasks in the workflow. These results in PSO based algorithm giving lower cost of execution as compared with BRS based algorithm. On the other hand, PSO based algorithm only considers "ready" tasks during scheduling iteration and cannot get the global optimal solution. So, RDPSO get better cost value than PSO based algorithm.

TABLE III. TOTAL COST FOR DIFFERENT SIZE OF DATA

Data size(M)	64	128	256	512	1024	2048
RDPSO	56.7	61.2	68.6	89.9	118.3	288.4
PSO	80.41	86.23	93.4	181.5	246.5	643.7
BRS	259.5	488.68	904.3	2107.1	4342.7	7843.6

In order to further study the optimization ability among the mentioned three algorithms, more complex workflow applications are involved. Fig 3 plots the total makespan optimization ratio of the three algorithms. From Fig 3, we can see that BRS can get around 2% optimization ratio, PSO can achieve from 6% to 8% optimization ratio, RDPSO can get from 10% to 17% optimization ratio on the whole makespan. PSO does not take makespan into account when it evolves; RDPSO takes not only computation cost but also whole makespan into account when it evolves. When user's requirement is specified, complete the workflow application with the requirement constraint is very important, so RDPSO is more applicable in cloud environment than PSO.

We also compared the total computation cost optimization ratio by varying the tasks number. The result is plotted in Fig 4. From Fig 4, we can see that both PSO and RDPSO can achieve relatively large optimization ratio. These two algorithms take cost into account while they are searching the optimal solutions. BRS only blindly choose the best service.

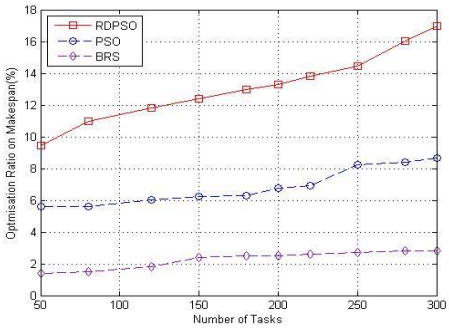

Figure 3. The total makespan optimization ratio

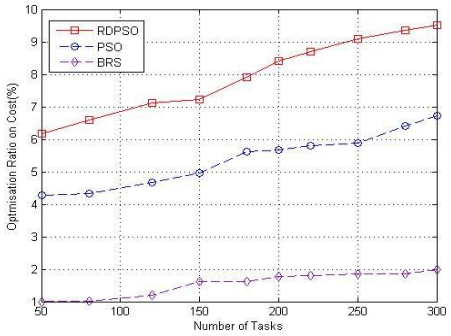

Figure 4. The total computation cost optimization ratio

On the other hand, the conclusion we can draw from Fig. 4 is that when the task number of the workflow becomes large, RDPSO optimization ratio increases relatively dramatic. It means RDPSO can achieve lower cost for executing the workflow. Take an example, when the task number is 250, RDPSO can get more than 9% cost reduction while PSO only get 6% cost reduction. So, when schedule the large scale workflow application in cloud computing environment, RDPSO is more promising than PSO.

V. CONCLUSION REMARKS

This paper presents a revised discrete particle swarm optimization algorithm to optimize the schedules of workflow application in cloud computing environment. In this algorithm, the candidate solution is presented by the set of task-service pairs, each particle not only learns from different exemplars, but also learns the other feasible pairs for different dimensions. The constructive position building procedure guarantees each position is feasible. This scheme greatly reduces the search space and enhances the algorithm performance. Based on the simulation results, the new algorithm yields outstanding performance on scheduling workflow applications in cloud environment.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments which are helpful to improve the presentation of the manuscript. This work is partially supported by the National Natural Science Foundation of China project (Grant No. 70871033).

REFERENCES

[1] B. Raghavan, et al., "Cloud control with distributed rate limiting," Proc. *SIGCOMM'07,* pp. 337 - 348, Kyoto, Japan, 2007.
[2] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," IEEE Transactions on Software Engineering, pp. 369-384, 2007.
[3] K. Bhattacharya, et al., "ICSE Cloud 09: First international workshop on software engineering challenges for Cloud Computing," Proc. 31st International Conference on Software Engineering - Companion Volume,. (ICSE-Companion 2009), pp. 482-483. 2009
[4] W. Van der Aalst and K. Van Hee, Workflow management: models, methods, and systems: The MIT press, 2004.
[5] I. Foster, Zhao Yong, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared", Proc. Grid Computing Environments workshop, 2008. GCE '08, pp. 1-10, 2008.
[6] D. Yuan, et al., "A data placement strategy in scientific cloud workflows," Future Generation Computer Systems, pp. 1200-1214 2010.
[7] S. Pandey, et al., "A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments," in Advanced Information Networking and Applications (AINA), 24th IEEE International Conference on, pp. 400-407,2010.
[8] J. Kennedy and R. Eberhart, "Particle swarm optimization," Pro. The IEEE International Conference on Neural Networks, pp. 1942-1948, Perth, Australia, 1995.
[9] D. Bratton and J. Kennedy, "Defining a Standard for Particle Swarm Optimization," in Swarm Intelligence Symposium, 2007. SIS 2007. IEEE, 2007, pp. 120-127.
[10] M. Clerc, "Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem," New optimization techniques in engineering(Springer), 2004.
[11] G. Pampara, et al., "Combining particle swarm optimisation with angle modulation to solve binary problems," Proc.The IEEE Congress on Evolutionary Computation , pp. 89-96, vol.1,2005.
[12] D. Sha and C. Hsu, "A hybrid particle swarm optimization for job shop scheduling problem," Computers & Industrial Engineering, pp. 791-808, vol. 51,2006.
[13] J. Grobler, et al., "Metaheuristics for the multi-objective FJSP with sequence-dependent set-up times, auxiliary resources and machine down time," Annals of Operations Research, pp. 1-32, 2008.
[14] M. Neethling and A. P. Engelbrecht, "Determining RNA Secondary Structure using Set-based Particle Swarm Optimization," IEEE Congress on Evolutionary Computation, BC, Canada,pp. 1670-1677, 2006.
[15] C. Wei-Neng, et al., "A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems," IEEE Transactions on Evolutionary Computation, , vol. 14, pp. 278-300, 2010.
[16] Z. Wu, et al., "A Market-Oriented Hierarchical Scheduling Strategy in Cloud Workflow Systems," Journal of Supercomputing, Special issue on Advances in Network&Parallel Comptg, to be appeared,2010.
[17] J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing," Journal of Grid Computing, no. 3, pp. 171-200, 2005.
[18] X. Liu, J. Chen, Z. Wu, Z. Ni, D. Yuan, Y. Yang, Handling Recoverable Temporal Violations in Scientific Workflow Systems: A Workflow Rescheduling Based Strategy. Proc. of 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid2010), pages 534-537, Melbourne, Australia, May 2010.