2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Pareto Simplified Swarm Optimization for Grid-computing Reliability and Service Makspan in Grid-RMS

Shang-Chia Wei, Wei-Chang Yeh, *Senior Member*, *IEEE*, and Tso-Jung Yen

***Abstract*—In a grid-computing service, Grid-RMS must generate suitable assignment combinations (execution blocks) for dependable service quality and satisfactory makespan (service time). In this paper, service reliability of a grid environment and makespan of a grid application are estimated via the universal generating function methodology and probability theory. Then, we represent a simplified swarm optimization (SSO) with the Pareto-set cluster (PC) to search the best assignment combinations in a grid environment with star topology. In terms of the task partition and distribution for a grid application, we employ a Pareto-set cluster to guide particle evolution, an elitist strategy to promote solution quality, and a simplified update mechanism to enhance the multi-objective optimization effectiveness. Finally, we assess the performance of the PC-SSO by the interactive tradeoff problem based on the analysis of four scenarios with respect to the bi-objective problem and given restrictions.**

I. Introduction

A grid environment consists of many distributed and heterogeneous computing devices, which can process several mass and complex workloads. Owing to the high-performance computing (HPC) service of a grid-computing cluster, the grid environment for all applications has a great future. However, in a grid environment, grid resource management system (Grid-RMS) usually requires efficiently assigning subtasks (task partitions) to adequate resources through the communication links. In recent decades, the widespread deployment of grid environments has become necessary for scientific and industrial applications [1]. Grid-computing, which involves virtual services generated from grid-computing environment, was proposed in 1995 in the I-WAY project by DeFanti et al. [2]. In brief, the grid environment connects idle computing devices on the network and clusters them together to provide high-performance computing (HPC) in a manner analogous to parallel processing. In fact, the grid environment is a distributed and heterogeneous computing environment. The grid-computing technique can facilitate resource virtualization, on-demand provisioning, and service (resource) sharing between organizations [3]. Thus, the grid computing not only provides collaborative resource sharing but also supplies high-performance computing services from dynamic virtual organizations at different locations.

The high performance of grid-computing facilitates the resolution of large-scale, highly complex, and time-consuming problems that can be processed solely by supercomputers. Examples of such problems include earthquake simulation, climate modeling, the Human Proteome Folding Project, and the Clean Energy Project. However, the performance of grid-computing techniques is a focal point for expanding applications of the grid environment. The grid environment using a resource management system (Grid-RMS) to operate the progress of task request assignment, when a Grid-RMS receives a task from job request of a client (Grid user), the Grid-RMS considers how to segment the task and how to allocate subtasks to suitable computing devices. The designated computing devices with assigned subtasks constitute an execution block (EB) for the Grid-RMS, and every EB can perform assigned subtasks simultaneously, in a manner analogous to parallel computing. After the designated computing devices finish their allocated subtasks in each EB, they return the result to the Grid-RMS, which then integrates all results into a complete output for the task and delivers this output to the client that requested the service. Hence, the performance of grid-computing techniques in terms of processing time and completive reliability is controlled by a Grid-RMS within the grid environment [4].

At present, the technique of grid-computing is still under development. One difficulty involves managing task partitions and devices allocation for large-scale applications of enterprises and governments. If the Grid-RMS of the grid environment is not competent to distribute adaptable computing devices to subtasks, the performance of the entire grid environment, including system reliability, processing time, and operational cost, would be affected. In grid-computing management issues such as these, no adequate methodology yet exists to handle the problem of resource (computing devices) and subtask allocation.

According to reviews about grid-computing [1]-[5], there are many essential QoS (quality of service) requirements for a grid-computing service, and these depend on the architecture of the grid environment. Without the essential services, the grid-computing technique would be hard to popularize. The global requirements that grid-computing must satisfy are listed below:

1) Reliability: The grid environment must promise continuous service, stable connections, and safe sharing.

2) Performance: To achieve data and resource sharing,

S.-C. Wei is with the Institute of Statistical Science, Academia Sinica, Taipei, Taiwan 115, R.O.C (corresponding author: phone: +886-02-2787-1955; e-mail: wsc@stat.sinica.edu.tw).

W.-C. Yeh is with the Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu, Taiwan 300, R.O.C (e-mail: yeh@ieee.org).

T.-J. Yen is with the Institute of Statistical Science, Academia Sinica, Taipei, Taiwan 115, R.O.C (e-mail: tjyen@stat.sinica.edu.tw)

the grid environment must make sufficient use of all computing resources to avoid wastage due to idle resources. Consequently, resource sharing should increase the effectiveness of the grid environment.

3) Standardization: Grid environment has a standard portal with normal interface and consistent protocol without involving restrictions on connection type or communication facility.

4) Accessibility: To request a service from a grid-computing system, clients must be able to connect arbitrarily on the network and to communicate with a browser in the customary manner at any time and from any location.

The extension of grid-computing will be in proportion to its implementation of the abovementioned requirements. Assuming correct standardization and normal accessibility in a secure grid environment, we focus herein on the optimization of two themes: service reliability and service time. Generally, we evaluate service reliability by analyzing assigned subtasks and designated online resources. However, different computing resources and communication links have different rates of execution and transmission failures when running EB subtasks for a grid environment. Thus, service reliability depends on the resources and links. Similarly, the resources and links of a grid environment affect the execution time as well because of their different computing and transmission speeds. Therefore, service reliability and execution time are strongly interrelated. In this paper, we propose using Pareto-set cluster Simplified Swarm Optimization (PC-SSO) to optimize the twin targets of service reliability and service time for a grid environment with a star topology.

The remainder of the paper is arranged as follows: In Section 2, we discuss the related work about the Grid-RMS. In Section 3, we describe the mathematical model for a grid environment with the star topology. In Section 4, we propose simplified swarm optimization with the concept of the Pareto-set cluster and elitist strategy. In Section 5, and we present and explain the optimization results of PC-SSO. Section 6 gives the advantages of using the PC-SSO method to address the bi-objective problem posed by grid environments, presents the results of the study, and mentions our future work.

II. Related Work

In this section, we discuss some related works about Grid-RMS first, and then point out the main diversity of our work. Generally, the aspect of Grid-RMS is analogous to a matchmaker or broker for subtask and resource matchmaking, which would dispose adequate assignment combinations for some certain quality of service (QoS) requirements like service reliability, makespan, and resource utility etc [6]. These feasible assignment combinations consisted of many task partitions and computing resources are execution blocks (EBs) for completing a grid application from a task request. Hence, the Grid-RMS is responsible for balancing supply-demand matchmaking process and generating subtasks-resources allocation proposals in a grid environment [7]. Although some Grid brokers have been developed for the basis of grid market and deployed in grid infrastructures, these brokers in consideration of fault tolerance, cost and runtime is still in its infancy [7], [8]. Buyya et al. [8], [9] developed a distributed computational economy-based resource broker, named Nimrod-G, to allocate and regulate supply and demand of a grid environment with a view of deadline, budget, and other required QoS. Dogan and Ozguner [10] studied that as the resource price could vary with different scheduling time intervals, the independent tasks would be scheduling towards multiple QoS requirements such as timeliness, reliability, version, and priority. Li and Li [11] proposed a price-directed proportional resource allocation algorithm for solving the grid task agent resource problem. When given specified completion time or resource pricing policy, this algorithm could spend the least possible amount of money or runtime to finish the grid application. Liu et al. [12] proposed a fuzzy particle swarm optimization to dynamically schedule the task requests to accomplish the minimal makespan. Chauhan and Joshi [13] published QoS Guided heuristic algorithms for grid task scheduling, whose simulation results using GridSim are better than other conventional algorithms. Yang et al. [14] proposed a Bayesian optimization algorithm to minimize the makspan for the multiprocessor scheduling problem in heterogeneous computing environments. In terms of a grid application and a grid environment, Dai and Levitin [5], [15]-[17] formulated a task allocation model and applied genetic algorithm to optimize a single objective function (makespan or reliability) with constraints. We looking ahead, it is necessary and predictable to simultaneously concern makespan, reliability and other QoS requirements in a Grid-RMS. Above related work mostly focused on a single objective function with additional constraints, which is not enough to fulfill important QoS requirements towards a distributed and heterogeneous computing environment. Furthermore, so far none really multi-objective evolutionary algorithm has been developed for the Grid-RMS, whose task allocation model is related to makespan of a grid application and service reliability of a grid environment. Chitra et al. [18] integrated hybrid local search with multi-objective evolutionary algorithm (MOEA) [19] such as SPEAII and NSGAII to optimize makespan and reliability index for the task scheduling system model; however, their assumptions of perfect communication link confined their work to a heterogeneous computing environment. Therefore, in terms of a distributed and heterogeneous computing environment, we first propose a bi-objective evolutionary algorithm (PC-SSO) as a task allocation model of Grid-RMS to deal with the interactive tradeoff problem between service reliability and makespan and to generate a non-dominated solution set so as to make task allocation by way of choosing a proper solution for a specified scenario.

III. Mathematical Formulation

In this paper, the deployment of grid environment we concentrated is a star topology depicted as Fig. 1 [5], where a computer host as a Grid-RMS to serve grid users is connected

computing nodes (resources) through their mutually independent communication channels (such as a Gigabyte Ethernet with star topology).

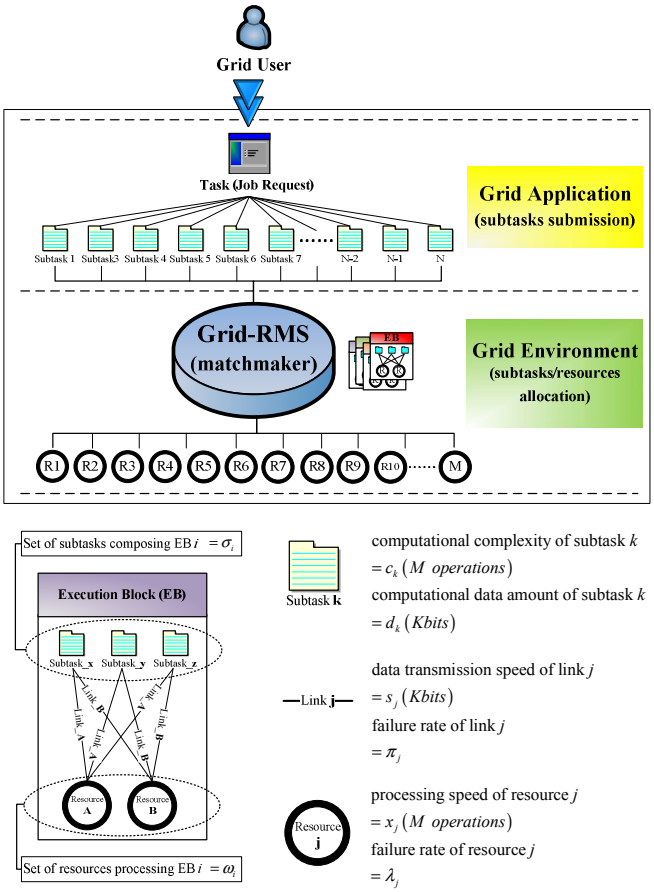

Fig. 2. Diagram of execution block in Grid-RMS

When the Grid-RMS receiving and approving one or more task request, it could perceive the idle resources among the provision agreement of the underlying grid-computing resources, and generated several execution blocks (EBs) composed by resources of a grid environment and subtasks of a grid application. In this paper, the execution blocks distributed and generated by Grid-RMS are shown in Fig. 1. The Grid-RMS [5] determines the subtasks and resources allocation, which forms a group within the EB. In each EB, every assigned subtask has a definite required batch of input data with fixed complexity (c_k) and a definite output data quantity (d_k). Similarly, the designated resources and respective connections of every EB possess individual processing speeds (x_j), processing failure rates (λ_j), transmission speeds (s_j), and communication failure rates (π_j). Thus, the Grid-RMS can estimate the service reliability and service time for a given set of EBs. In other words, the service reliability and service time can vary between unequal EBs with different subtask sets (σ_i) and resource sets (ω_i).

In terms of the task allocation model of a Grid-RMS [5], The Grid-RMS divides tasks requested by clients into m subtasks that can be executed independently by resources within the grid environment. Thus, the computational complexity Ci of EB-i, which comprises a set of subtasks σi, is

$$C_i = \sum_{k=1}^{m} c_k \cdot 1(k \in \sigma_i) \tag{1}$$

Grid-RMS sends essential data blocks to a resource of EB-i, and the resources of EB-i also send generated data to the Grid-RMS. The total amount Di of input and output data transmitted between the Grid-RMS and an EB-i resource is given by

$$D_i = \sum_{k=1}^{m} O_k \cdot 1(k \in \sigma_i) + \sum_{x=1}^{n} I_x \cdot 1\left(x \in \bigcup_{k \in \sigma_i} B_k\right), \tag{2}$$

where B_k denotes a set of required input data blocks for subtask k, I_x denotes the size of data block x, and O_k denotes the amount of output data produced for subtask k by a resource. Moreover, the Grid-RMS does not perform multiple imports of the identical input-data block *x*; therefore, the total amount of input data for EB-i diminishes by removing identical data blocks from different subtask k of EB-i via the union operation ($\bigcup_{k \in \sigma_i} B_k$). Therefore, the total amount of data transmitted from the Grid-RMS to an EB-i resource is $\sum_{x=1}^{n} I_x \cdot 1\left(x \in \bigcup_{k \in \sigma_i} B_k\right)$. In EB-i, the execution state of resource j processing a set of subtasks σ_i may be either complete or incomplete. The random time t_{ij} to complete subtasks σi can be estimated by

$$t_{ij} = \begin{cases} \hat{t}_{ij} = \dfrac{C_i}{x_j} + \dfrac{D_i}{s_j} & \text{, if } \sigma_i \text{ has been done by R - } j \\ \infty & \text{, if } \sigma_i \text{ has not been done by R - } j \end{cases}, \tag{3}$$

The time for incomplete subtasks σ_i is infinity.

If the resource j and link j do not fail during the time that subtasks σ_i are processed by resource j (in EB-i), the probability of completing the subtasks is

$$\Pr\left(t_{ij} = \hat{t}_{ij}\right) = p_j\left(\hat{t}_{ij}\right) = e^{-\left(\lambda_j + \pi_j\right)\hat{t}_{ij}}, \tag{4}$$

and the probability that subtasks σi are incomplete is the following complement:

$$\Pr\left(t_{ij} = \infty\right) = 1 - p_j\left(\hat{t}_{ij}\right) \tag{5}$$

In EB-i, an individual resource j from among a set of resources ω_i is allocated identical required processing subtasks σ_i. Thus, the random time to complete EB-i is the shortest time within which subtasks σ_i can be processed by resource j from among the set of resources ω_i:

$$\theta_{i,\omega_i} = \min_{j \in \omega_i}\left(\hat{t}_{ij}\right) \tag{6}$$

In the grid environment, the entire task is finished when all the subtasks of EB-i are finished. Thus, the random time for completing the entire task is

$$\Theta = \max_{1 \le i \le h} \theta_{i,\omega_i} = \max_{1 \le i \le h}\left[\min_{j \in \omega_i}\left(\hat{t}_{ij}\right)\right] \tag{7}$$

Levitin and Dai [5] use the u-function U_h, [20]-[22] for the PMF of the service time Θ of a grid environment with star toplogy, so the service reliability [$R(\Theta \neq \infty)$] as well as the conditional expected service time W can be estimated. After the u-function $U_h(z)$ eliminates the probability of incomplete grid application ($\Theta = \infty$), the probabilities of completion grid

application are summed as the service reliability [$R(\Theta \neq \infty)$]:

$$R(\Theta \neq \infty) = \Pr(\Theta \leq \theta^* = \infty)$$
$$= \sum_{f=1}^{F} \Pr(\Theta = \Theta_f) \cdot 1(\Theta \leq \theta^*) = \sum_{f=1}^{F} Q_f \cdot 1(\Theta \leq \theta^*) \quad (8)$$

The conditional expected service time W of the grid environment can be evaluated through the conditional expected value $E(\Theta \mid \Theta \neq \infty)$ as follows:

$$W = E(\Theta \mid \Theta \neq \infty) = \sum_{f=1}^{F} \Theta_f \cdot \Pr(\Theta_f \mid \Theta \leq \theta^* = \infty)$$
$$= \sum_{f=1}^{F} \Theta_f \cdot \frac{\Pr(\Theta_f)}{\Pr(\Theta \leq \theta^*)} = \left(\sum_{f=1}^{F} \Theta_f \cdot Q_f\right) \Big/ \left(R(\Theta \leq \theta^*)\right) \quad (9)$$
$$= \left(\sum_{f=1}^{F} \Theta_f \cdot Q_f\right) \Big/ \left(\sum_{f=1}^{F} Q_f \cdot 1(\Theta \leq \theta^*)\right)$$

Thus, the resource allocation model for a Grid-computing service can be formulated as a discrete bi-objective COP (combinatorial optimization problem), for which the decision variables and bi-objective function are described as follows:

Decision variables

Binary variables: $\mathbf{X}=\{x_{kj} \mid x_{kj}=0 \text{ or } 1,\ k=1,\ldots,N,\ j=1,\ldots,M\}$, where x_{kj} presents the state of subtask k assigned on resource j. If subtask k is processed by resource j, the x_{kp} is 1, otherwise the x_{kp} is 0. The solution space is $2^{N\times M}$.

Objective function

Maximize $R(\Theta \neq \infty)$

Minimize W

Subject to:

$$1 \leq \sum_{j=1}^{M} x_{kj} \leq M,\ \forall\ k, \quad (10)$$

$$\Theta \leq \theta^*, \quad (11)$$

where θ^* is the upper bound of service-time, that is, the guaranteed service-time as a QoS term for the SLA.

IV. Methodology

For the grid-computing reliability-time tradeoff problem, we propose a Pareto-set cluster simplified swarm optimization (PC-SSO) to search the (near) Pareto solutions with the Pareto frontier in grid environments with star topologies. The SSO was originally designed by Yeh [23]-[25] in 2009 to overcome the drawback of PSO in discrete problems and was called discrete PSO (DPSO). Simulation results reveal that SSO converges faster and offers a higher-quality solution than PSO alone. The procedure of the PC-SSO is illustrated in Fig. 2.

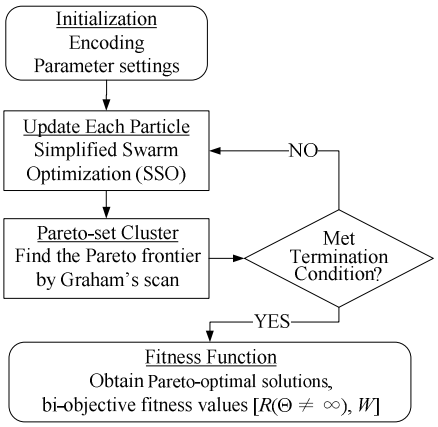

Fig. 2. Flowchart for Pareto-set cluster simplified swarm optimization (PC-SSO)

For the max-min bi-objective problem, Pareto frontier consists of several pairs of non-dominated bi-objective fitness values [$R(\Theta\neq\infty)$, W] that originate from one or more non-dominated subtasks-resources allocations. To find the Pareto frontier for each iteration, we adopt Graham's scan [26], [27] to find a convex hull from which we construct several line segments to P1 (W_{min}, R_{min}) and P5 (W_{max}, R_{max}). We connect the provisional Pareto points resulted from convex hull, and then draw a Pareto frontier. Line segments LS of the convex hull CH and a set of Pareto points Pt are depicted in Fig. 3.

Afterwards, we develop a concept of the Pareto-set cluster as the evolutional guidance of Pbest and Gbest, because Pbest and Gbest cannot determine the quality and diversity in the bi-objective problem. Therefore, we combine the concept of the Pareto-set cluster with SSO to find the Pareto frontier. The Pareto-set cluster is some part of Pareto points (Pt) wrapped in line segment (LS). In other words, the concept of Pareto-set clusters is used to group non-dominated solutions into several clusters to guide particle evolution so that each particle has a limited division to explore. Pareto-set cluster ($PC_{c=4}$) includes Pareto points P_4, D_1, D_2, P_5 as shown in Fig. 3. For example, we have obtained a provisional Pareto frontier (PF) at an iteration, so the PF = {P_1, A_1, P_2, B_1, B_2, P_3, C_1, C_2, P_4, D_1, D_2, P_5}. If we let LS_1 ={P_1, P_2 }, LS_2 ={P_2, P_3 }, LS_3 ={P_3, P_4 }, LS_4 ={P_4, P_5}, then we obtain PC_1 ={P_1, A_1, P_2 }, PC_2 = { P_2, B_1, B_2, P_3 }, PC_3 = {P_3, C_1, C_2, P_4}, PC_4 = { P_4, D_1, D_2, P_5}.

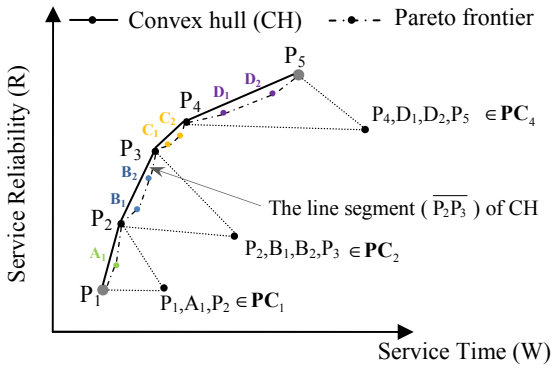

Fig. 3. Conceptual diagram of Pareto frontier (PF) and Pareto-set cluster (PC)

After having obtained PF and PC, we can determine the selection probability Pr(PC) for each Pareto-set cluster for the next iteration. The sum dist(PC_c) gives the sum of the distance between two ordered and adjoined points in a Pareto-set cluster PC_c. The sum dist(PF) gives the sum of the distance between two ordered and adjoined points in a set of Pareto points PF. Therefore, Pr(PC_c) is $\Sigma dist(PC_c)$ divided by $\Sigma dist(PF)$. In the next iteration, a particle can choose a Pareto-set cluster PC on which it can evolve based on the selection probability Pr(PC) of each Pareto-set cluster. In Fig. 3, the selection probability of Pareto-set cluster Pr(PC_4) is calculated from the sum ($\overline{P_4D_1} + \overline{D_1D_2} + \overline{D_2P_5}$) divided by the sum ($\overline{P_1A_1} + \overline{A_1P_2} + \overline{P_2B_1} + \overline{B_1B_2} + \overline{B_2P_3} + \overline{P_3C_1} + \overline{C_1C_2} + \overline{C_2P_4} + \overline{P_4D_1} + \overline{D_1D_2} + \overline{D_2P_5}$) after the scales in service reliability and service time [$R(\Theta\neq\infty)$, W] are standardized.

Furthermore, we apply the elitist strategy to PC-SSO, which generates a new Pareto frontier produced by integrating the previous and present Pareto frontiers, except for the 1st generation, and eliminates particles far from the convex hull. Thus, the elitist strategy can guarantee amelioration of the convex hull for each subsequent generation simply by retaining positive solutions and discarding negative ones.

When updating each particle, PC-SSO takes the Pareto-set cluster PC and its selection probability Pr(PC) as the evolution trend for each particle upon the next generation. First, two particle *pca* and *pcb* randomly selects from a Pareto-set cluster PC by using the selection probability Pr(PC_c) of each Pareto-set cluster. There are four roles (*pca*, *pcb*, current solution, random solution) in SSO. At iteration t, *pca* and *pcb* are two solutions of a Pareto-set cluster PC in the best-so-far PF, the current solution is x_i, and the random solution is a solution x_r generated at random. Subsequently, the updated mechanism is responsible for the process of each particle. The updated mechanism of the PC-SSO is based on the following simple mathematical model after c_w, c_p and c_g are given:

$$x_i^t = \begin{cases} x_i^{t-1} & ,if\ r_i^t \in [0, C_w) \\ pca_i & ,if\ r_i^t \in [C_w, C_p) \\ pcb_i & ,if\ r_i^t \in [C_p, C_g) \\ x_r & ,if\ r_i^t \in [C_g, 1) \end{cases}, \quad (12)$$

where $C_p = c_p + c_w$, and $C_g = c_g + c_p + c_w$.

V. Experiment Result

A. Model of Bi-objective Problem with No Restrictions

We adopted a grid-computing instance [5] to validate the optimization effectiveness of PC-SSO. The grid environment holds 6 resources and 8 independent subtasks. In the initial iteration, the PC-SSO randomly selects 1000 particles. We observe that these particles are scattered over the solution space rather than gathered in one region, some of which are located far from the ideal corner (R_{max}= 1, W_{min}= 0). Because PC-SSO retains elite particles and diminishes passive particles, all particles move toward the latest Pareto frontier. Positive particles in each generation are retained and updated (Fig. 4).

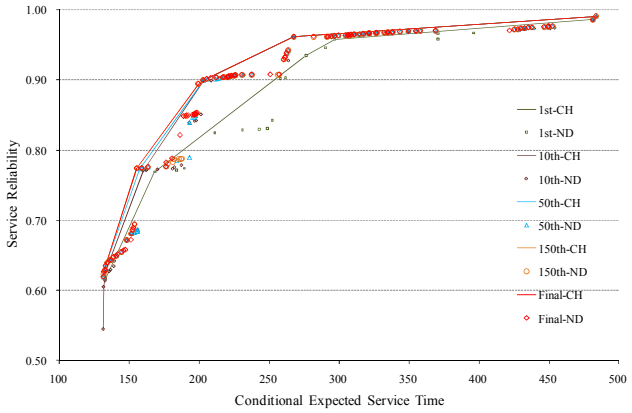

Fig. 4. Diagram of convex hull (CH) and non-dominated solutions (ND equal to PF) of every generation.

By repeating 30 calculations of PC-SSO with identical setting 600 generations and 1000 particles, we obtain the best Pareto frontier PF. The best PF is listed in Table I. The best PF has 122 pairs of objective values [W, $R(\Theta\neq\infty)$], which are discrete and piecewise distributed in solution space. However, if we randomly run a calculation of PC-SSO with an identical setup, we can obtain at least 100 Pareto optimal solutions, and a final PF could revive 71% (± 5%) of the best PF. In other words, we acquire 86 of the best PF, and the other solutions of a final PF closely approximate the solutions of the best PF. Furthermore, the best PF, whose service reliability is greater than 0.9, is repeated at a rate of 92% (± 4%) by a final PF, which means that the final PF can duplicate 62 of the best PF with a service reliability of 0.9.

TABLE I
THE BEST PARETO FRONTIER PF=[W, $R(\Theta\neq\infty)$]

W	R	W	R	W	R	W	R	W	R	W	R
131.29	0.61911	151.32	0.69122	188.10	0.84865	222.81	0.90544	296.68	0.96280	349.63	0.96932
131.44	0.62000	151.41	0.69631	190.15	0.84983	222.85	0.90578	299.44	0.96330	354.93	0.96963
131.50	0.62632	152.52	0.77366	191.07	0.84986	224.72	0.90585	304.64	0.96335	358.02	0.96995
132.38	0.62775	155.20	0.77417	193.92	0.85022	225.19	0.90648	306.15	0.96362	369.13	0.97018
132.62	0.62845	158.69	0.77431	194.23	0.85042	225.98	0.90687	306.66	0.96380	421.94	0.97023
132.81	0.63470	158.73	0.77435	195.54	0.85157	230.49	0.90718	308.24	0.96400	425.36	0.97189
134.10	0.63914	159.18	0.77470	195.93	0.85166	233.56	0.90738	308.39	0.96443	427.88	0.97195
135.60	0.64263	162.78	0.77490	198.00	0.85288	237.44	0.90758	310.54	0.96464	430.05	0.97348
136.75	0.64300	163.16	0.77573	199.00	0.89439	240.29	0.90776	310.87	0.96495	432.62	0.97356
137.59	0.64462	168.86	0.77619	199.63	0.89456	257.16	0.90821	314.50	0.96499	435.43	0.97469
138.48	0.64715	174.47	0.77630	202.40	0.89911	260.33	0.92882	317.20	0.96566	438.08	0.97479
139.75	0.65033	174.57	0.77732	202.63	0.89946	261.23	0.93188	321.12	0.96634	446.71	0.97550
140.04	0.65218	175.66	0.78205	205.19	0.90130	261.63	0.93333	321.67	0.96641	449.54	0.97567
142.57	0.65335	176.16	0.78207	207.97	0.90273	262.32	0.93809	326.38	0.96692	449.85	0.97644
144.26	0.65754	176.61	0.78212	212.02	0.90358	263.40	0.94212	326.95	0.96717	452.69	0.97662
146.88	0.65817	176.71	0.78219	216.52	0.90365	264.26	0.94439	332.41	0.96730	481.52	0.98589
147.12	0.66979	177.74	0.78312	217.46	0.90374	267.48	0.96155	334.42	0.96740	483.84	0.99091
148.05	0.67132	180.39	0.78785	218.04	0.90389	279.59	0.96161	335.09	0.96769	–	–
149.51	0.67209	180.81	0.78790	220.48	0.90438	291.11	0.96169	338.03	0.96772	–	–
149.53	0.67905	181.94	0.78891	220.71	0.90475	292.08	0.96199	338.20	0.96821	–	–
151.21	0.68939	185.55	0.82169	221.23	0.90511	294.44	0.96206	343.49	0.96890	–	–

In addition, each service-time PMF of all Pareto solutions

for the best Pareto frontier PF can be derived by the UGFM, which can assist us in knowing about PF data such as the minimum service time, maximum service time, and the conditional expected service time (CEST equal to W) as well as its probability relative to individual completion service time. The procedure for obtaining a PMF is presented in the Appendix. Accordingly, we can determine a suitable allocation of subtasks and resources from the best PF for various scenarios. For example, if a grid-computing user is cautious with both completion service time and CEST, we can filter out some allocations from low-risk regions of PF because these regions have appropriate maximum completion time and small disparity with CEST. Conversely, if a user is just concerned about the optimization of two targets, high-risk as well as low-risk regions would be considered together. In this case, each PF service time is shown in Fig. 5.

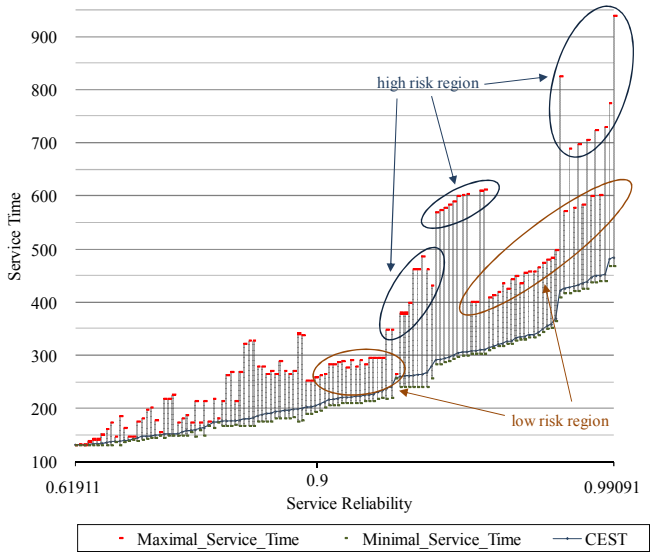

Fig. 5. Maximum, minimum, and conditional expected service time of Pareto frontier (PF = [W, $R(\Theta\neq\infty)$])

B. Model of Bi-objective Problem with Maximum Allowed Service Time

We now turn to an actual constraint—the maximum allowed service time θ^*. If a grid-computing user claims to restrain the maximum completion time, some Pareto frontiers PF = [W, $R(\Theta\neq\infty)$] would become infeasible solutions because their maximum completion time would exceed the maximum allowed service time. Thus, we transform the bi-objective problem [W, $R(\Theta\neq\infty)$] with the restriction of maximum allowed service time θ^* into another bi-objective problem [θ, $R(\theta^*)$].

By repeating 30 calculations of PC-SSO with identical setting 600 generations and 1000 particles, we obtain the best Pareto frontier PF = [θ, $R(\theta^*)$] in this study. The best PF is listed in Table II. In this study, the best PF has 114 pairs of objective values [θ, $R(\theta^*)$], which are discrete and piecewise distributed in solution space. There are at least 96 Pareto solutions in a PC-SSO experiment, and a final PF could revive 81% (± 4%) of the best PF. In other words, we acquire 92 of the best PSGS every time, and the other solutions of the final PF closely approximate solutions of the best PF. Furthermore, the best PF, whose service reliability is greater than 0.9, is repeated at a rate of 97% (± 3%) by the final PF, which means that the final PF can duplicate 66 of the best PSGS with a service reliability of 0.9.

TABLE II
THE BEST PARETO FRONTIER [PF = {Θ, $R_{max}(\Theta^*)$}] WITH THE RESTRICTION OF MAXIMUM ALLOWED SERVICE TIME Θ^*

θ	$R_{max}(\theta^*)$	θ	$R_{max}(\theta^*)$	θ	$R_{max}(\theta^*)$	θ	$R_{max}(\theta^*)$	θ	$R_{max}(\theta^*)$	θ	$R_{max}(\theta^*)$
131.29	0.61911	168.33	0.77280	231.75	0.89277	278.50	0.91077	353.00	0.95743	435.00	0.96740
131.50	0.62632	169.33	0.77368	232.25	0.89366	280.50	0.91168	357.00	0.95759	439.95	0.96760
135.25	0.62919	174.00	0.77490	233.50	0.89436	281.25	0.91213	362.33	0.95837	446.76	0.96817
136.00	0.63630	175.33	0.77619	234.25	0.89562	284.75	0.91411	368.33	0.95856	448.48	0.96818
138.57	0.64386	181.33	0.77732	235.75	0.89711	290.25	0.91454	376.67	0.95871	454.29	0.96849
141.86	0.65130	196.00	0.77838	243.25	0.89874	291.75	0.91470	378.00	0.96013	461.10	0.96864
146.33	0.65218	197.67	0.77860	245.50	0.89977	293.50	0.91552	378.38	0.96029	464.67	0.96890
146.67	0.65335	205.14	0.81415	248.50	0.90065	294.50	0.91602	378.67	0.96034	473.33	0.96932
148.43	0.65615	208.29	0.81726	249.50	0.90309	295.50	0.91641	380.67	0.96145	480.33	0.96963
149.57	0.65833	209.20	0.81785	257.75	0.90370	299.50	0.91712	385.62	0.96202	483.00	0.96995
150.67	0.67501	209.43	0.81812	259.75	0.90477	301.25	0.94459	392.43	0.96217	498.00	0.97018
152.57	0.67846	213.00	0.83404	262.00	0.90502	301.50	0.94525	393.71	0.96264	571.00	0.97189
153.57	0.67876	214.10	0.83448	262.25	0.90582	312.75	0.94577	394.14	0.96291	576.67	0.97348
155.33	0.67983	214.14	0.83478	262.75	0.90668	313.25	0.94605	399.67	0.96400	583.67	0.97469
157.14	0.68173	217.33	0.83535	264.25	0.90821	323.00	0.95600	399.95	0.96443	598.67	0.97550
158.29	0.76900	220.50	0.88926	271.00	0.90919	327.33	0.95622	408.19	0.96499	601.33	0.97644
160.67	0.77017	221.00	0.89179	271.75	0.90934	334.33	0.95628	413.14	0.96566	729.13	0.97662
162.67	0.77228	230.00	0.89221	275.25	0.91004	337.67	0.95728	418.95	0.96634	773.60	0.98589
165.67	0.77248	231.00	0.89260	276.00	0.91043	351.00	0.95741	425.76	0.96692	939.33	0.99091

On the basis of the Pareto frontier PF = [θ, $R(\theta^*)$], we can pick out an adoptable solution, including the maximum service reliability $R_{max}(\theta^*)$, to restrict the changes in the maximum allowed time θ^* (Fig. 6). Details of the Pareto frontier with a restricted maximum allowed service time [PF =θ, $R(\theta^*)$] are listed in Table 6.

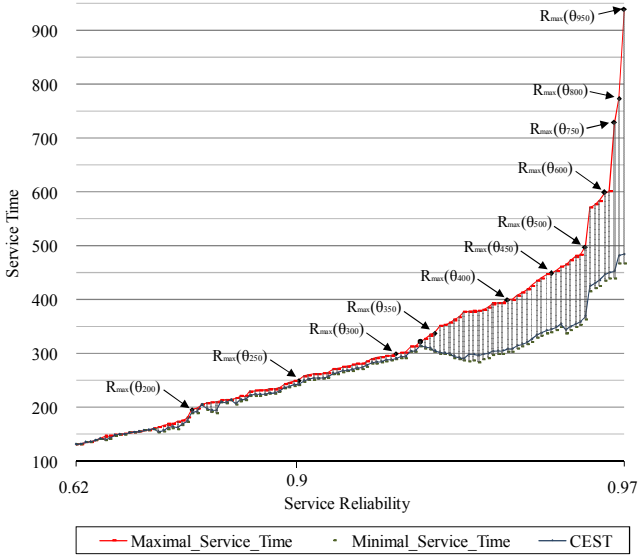

Fig. 6. Maximum, minimum, and conditional expected service time of Pareto frontier [**PF** = [θ, $R(\theta^*)$]| $\theta \leq \theta^*$}]

TABLE III
NON-DOMINATED SUBTASK OR RESOURCE ALLOCATIONS {SUBTASKS, RESOURCES} WITH RESTRICTED MAXIMUM ALLOWED SERVICE TIME Θ^*

θ^*	Θ_{min}	Θ_{max}	$R_{max}(\theta^*)$	W
200	189.429	197.667	0.77860	191.624
	{6,**R5**}{4,7,**R3,R4**}{1,2,**R1,R2**}{3,5,8,**R6**}			
250	241.143	249.500	0.90309	242.261
	{4,7,**R2,R3**}{2,6,**R1,R4**}{1,3,5,8,**R5,R6**}			
300	289.000	299.500	0.91712	290.615
	{2,4,**R2,R3**}{7,**R1**,**R4**}{1,3,5,6,8,**R5,R6**}			
350	303.286	337.667	0.95728	305.261
	{2,6,7,**R1,R2,R3,R4**}{1,3,4,5,8,**R5,R6**}			
400	303.286	399.952	0.96443	308.392
	{2,6,7,**R1,R2,R3**}{1,3,4,5,8,**R4,R5,R6**}			
450	340.143	448.476	0.96818	344.797
	{3,**R1,R2,R3**}{1,2,4,5,6,7,8,**R4,R5,R6**}			
500	363.429	498.000	0.97018	369.129
	{5,7,**R1,R2,R4**}{1,2,3,4,6,8,**R3,R5,R6**}			
600	436.571	598.667	0.97550	446.710
	{8,**R1,R2**}{1,2,3,4,5,6,7,**R3,R4,R5,R6**}			
750	439.857	729.133	0.97662	452.685
	{5,**R1,R4**}{1,2,3,4,6,7,8,**R2,R3,R5,R6**}			
800	466.857	773.600	0.98589	481.518
	{**R1**}{1,2,3,4,5,6,7,8,**R2,R3,R4,R5,R6**}			
1000	466.857	939.333	0.99091	483.838
	{1,2,3,4,5,6,7,8,**R1**,**R2,R3,R4,R5,R6**}			

If the constraint of minimum satisfied service reliability R^* is set to 0.9, we must leave solutions of reliability greater than 0.9 in the best PF (Fig. 5) because we still focus on the bi-objective problem (W, $R(\Theta\neq\infty)$). However, when two fixed constraints exist for the maximum allowed service time θ^* and the minimum satisfied service reliability R^*, we still can optimize another model of the bi-objective problem [θ, $R(\theta^*)$] and select the non-dominated solutions that satisfy the maximum allowed service time and minimum satisfied service reliability from the best PF (Fig. 6). Consequently, PC-SSO is sufficiently efficient to solve the four types of bi-objective problems with fixed restrictions discussed in this paper.

VI. CONCLUSION

In this paper, we propose a new bi-objective evolutionary algorithm, Pareto-set Cluster Simplified Swarm Optimization (PC-SSO), to solve the interactive tradeoff problem between service reliability and service makespan in a grid environment with star topology. The reliability-time tradeoff problem is a nonlinear integer programming problem. We aimed to efficiently find many adoptable subtasks-resources allocations to achieve the best non-dominated service reliability and service time. In PC-SSO, we employed Pareto-set cluster elitist strategy and simplified the update mechanism to enhance its multi-objective optimization effectiveness. First, the concept of the Pareto-set cluster can group non-dominated solutions into several clusters to guide particle evolution while allowing each particle to explore a limited division. Second, the elitist strategy can retain positive solutions and discard negative ones. Third, by using the update mechanism inherited from SSO, PC-SSO is easy to implement, has few parameters (population and termination condition) to tune, and converges rapidly.

The optimization effectiveness of PC-SSO is validated through a numeric example of a grid application. The greater part of the Pareto frontier can be found in a PC-SSO calculation. No computation time of the PC-SSO exceeds 15 minutes Pentium 2.4 GHz with 4G RAM using VBA (Visual Basic for Applications) with 1000 populations and 600 generations. Moreover, we analyze four scenarios with respect to the bi-objective problem and restrictions and verify the optimization effectiveness of PC-SSO in the interactive bi-objective problem. Experiment results show that the Grid-RMS can make a suitable task allocation for a specified scenario from numerous non-dominated solutions. However, the use of a few resources can slightly boost service reliability because of their high processing failure rate, but their bandwidth and processing speed is lower than that of the other resources; therefore, the service time suddenly increases. This problem can be possibly solved in future studies by using the concept of unit cost or marginal utility

REFERENCES

[1] I. Foster and C. Kesselman, *The Grid2: Blueprint for a New Computing Infrastructure*, Morgan Kauffmann, 2nd edition, 2004.

[2] T. A. DeFanti, I. Foster, M. Papka, R. Stevens, and T. Kuhfuss, "Overview of the I-WAY: Wide Area Visual Supercomputing," *International Journal of Supercomputer Applications*, vol. 10, no. 2, pp. 123-130, 1996. -p1

[3] P. Plaszczak and R. Wellner, Jr., *Grid computing: The savvy manager's guide*, Morgan Kauffmann, 2006.

[4] K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of grid resource management system for distributed computing", *Software-Practice and Experience*, vol. 32, no. 2, pp. 135-164, 2002.

[5] G. Levitin and Y. S. Dai, "Optimal service task partition and distribution in grid environment with star topology", *Reliability Engineering and System Safety*, vol. 93, pp. 152-159, 2008.

[6] X. Bai, H. Yu, Y. Ji, and D. C. Marinescu, "Resource matching and a matching service for an intelligent grid", *International Journal of Computational Intelligence*, vol.1, no.3, pp.163-171, 2004.

[7] A. Clematis, A. Corana, D. D'Agostino, A. Galizia, and A. Quarati, "Job-resource matchmaking on Grid through two-level benchmarking", *Future Generation Computer Systems*, vol. 26, no. 8, pp. 1165-1179, 2010.

[8] D. Abramson, R. Buyya, and J. Giddy, "A computational economy for grid computing and its implementation in the Nimrod-G resource broker", *Future Generation Computer Systems*, vol. 18, no. 8, pp. 1061-1074, 2002.

[9] R. Buyya, D. Abramson, and J. Giddy, "Nimrod-G: an architecture for a resource management and scheduling system in a global computational grid", *International Conference on High Performance Computing in Asia-Pacific Region* (HPC Asia 2000), Beijing, China, IEEE Computer Society Press, USA, 2000.

[10] A. Dogan and F. Ozguner, "Scheduling independent tasks with QoS requirements in grid computing with time-varying resource prices", *In Proceedings of GRID 2002*, LNCS, vol.2536, pp.58-69, 2002.

[11] C. Li and L. Li, "Competitive proportional resource allocation policy for computational grid", *Future Generation Computer Systems*, vol. 20, no. 6, pp. 1041-1054, 2004.

[12] H. Liu, A. Abraham, A. Ella Hassanien, "Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm", Future Generation Computer Systems, vol.26, no.8, pp.1336-1343, 2010.

[13] S. S. Chauhan and R. C. Joshi, "QoS Guided Heuristic Algorithms for Grid Task Scheduling", *International Journal of Computer Applications*, vol. 2, no. 9, pp. 24-31, 2010.

[14] J. Yang, H. Xu, L. Pan, P. Jia, F. Long, M. Jie, "Task scheduling using Bayesian optimization algorithm for heterogeneous computing environments", Applied Soft Computing, 2011, doi:10.1016/j.asoc.2010.11.029.

[15] Y. S. Dai and G. Levitin, "Optimal Resource Allocation for Maximizing Performance and Reliability in Tree-Structured Grid Services," *IEEE Transactions on Reliability*, vol. 56, no. 3, pp. 444-453, 2007.
[16] Y. S. Dai and G. Levitin, "Optimal resource allocation for maximizing performance and reliability in tree-structured grid services," *IEEE Transactions on Reliability*, vol. 56, no. 3, pp. 444-453, 2007.
[17] Y. S. Dai, G. Levitin, and X. Wang, "Optimal task partition and distribution in grid service system with common cause failures," *Future Generation Computer Systems*, vol. 23, no. 2, pp. 209-218, 2007.
[18] P. Chitra, R. Rajaram, and P. Venkatesh, "Application and comparison of hybrid evolutionary multiobjective optimization algorithms for solving task scheduling problem on heterogeneous systems", *Applied Soft Computing*, vol. 11, no. 2, pp. 2725-2734, 2011.
[19] K. Deb, *Multi-objective Optimization Using Evolutionary Algorithms*, Wiley, New York, 2001.
[20] G. Levitin, *Universal Generating Function in Reliability Analysis and Optimization*, Springer-Verlag, 2005.
[21] A. Lisnianski and G. Levitin, *Multi-State System Reliability. Assessment, Optimization and Applications*, World Scientific, 2003.
[22] W. C. Yeh, "A Simple Universal Generating Function Method to Search for All MPs in Networks", *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans*, vol. 39, no. 6, pp. 1247-1254, 2009.
[23] W. C. Yeh, "A Two-Stage Discrete Particle Swarm Optimization for the Problem of Multiple Multi-Level Redundancy Allocation in Series Systems", *Expert Systems with Applications*, vol. 36, no. 5, pp. 9192-9200, 2009.
[24] W.C. Yeh, "Optimization of the disassembly sequencing problem on the basis of self-adaptive simplified swarm optimization," *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans*, vol. 42, no. 1, pp. 250-261, 2011.
[25] W.C. Yeh, "Simplified swarm optimization in disassembly sequencing problems with learning effects, *Computers & Operations Research*," vol. 39, no. 9, pp. 2168-2177, 2012.
[26] R. E. Steuer, *Multiple Criteria Optimization: Theory, Computation and Applications*. New York: Wiley, 1986.
[27] F. P. Preparata, M. I. Shamos. *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.