Multi-objective Planning for Workflow Execution on Grids

Jia Yu, Michael Kirley, and Rajkumar Buyya

Grid Computing and Distributed Systems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, VIC 3010 Australia
{jiayu,mkirley,raj}@csse.unimelb.edu.au

***Abstract*— Utility Grids create an infrastructure for enabling users to consume services transparently over a global network. When optimizing workflow execution on utility Grids, we need to consider multiple Quality of Service (QoS) parameters including service prices and execution time. These optimization objectives may be in conflict. In this paper, we have proposed a workflow execution planning approach using multi-objective evolutionary algorithms (MOEAs). Our goal was to generate a set of trade-off scheduling solutions according to the users QoS requirements. The alternative trade-off solutions offer more flexibility to users when estimating their QoS requirements of workflow executions. Simulation results show that MOEAs are able to find a range of compromise solutions in a short computational time.**

I. Introduction

Utility computing services has been reinforced by service-oriented Grid computing [1], which creates an infrastructure enabling users to consume services transparently over a secure, shared, scalable, sustainable and standard world-wide network environment. Utility computing [2] provides an economy model in which service providers are paid for their services according to factors such as the Quality of Service (QoS) provided.

Many applications in scientific and enterprise domains such as bioinformatics and financial analysis can be constructed as workflows. As a result, a number of Grid workflow management systems [3][4][5] have been developed to facilitate the composition and execution of workflow applications over distributed resources. Many heuristics [6][7][8] have also been proposed for workflow scheduling in order to optimize a single objective, such as minimizing execution time. However, a large number of objectives need to be considered when scheduling workflows on utility Grids based on users' QoS requirements. One important QoS metric is the processing power offered by the provider. Higher processing power means faster execution time and may result in a higher cost. Therefore, users may elect to "trade-off" between expensive services that may offer a faster execution time and cheaper but slower services to meet their requirements.

Recently, a number of cost-aware workflow scheduling heuristics [9][10] have been proposed and evaluated. Even though multiple criteria have been considered, their aim was to optimize a single objective. For example, they either minimize execution cost while meeting users' deadline or minimize execution time while meeting users' budget. However, in many situations users may need more information such as the range of QoS levels available and associated costs before making decisions. Furthermore, QoS levels and prices offered by service providers may be highly diverse and may not be directly correlated with the utility perceived by the users. For example, users may prefer some assignments which have slightly longer execution times but offer large savings in execution cost.

Given this motivation, we propose a workflow planning method, which considers simultaneously optimizing multiple objectives. In this paper, we investigate the trade-off between two conflicting objectives execution time and cost-while meeting the users' maximum deadline and budget requirements. We believe that our workflow planning approach can be easily extended to support more objectives. The major benefit of this workflow planning is to generate a set of alternative trade-off solutions and offer more flexibility to users for estimating their QoS requirements of workflow executions. It can also simultaneously generate the estimated sub-deadline and sub-budget of each single workflow task required for the desired schedule. Such information is important for establishing service level agreements between users and service providers for workflow task execution.

The remainder of the paper is organized as follows. In Section II, we introduce related work and compare them with the work proposed in this paper. We introduce the workflow planning problem in Section III. We present a general introduction to multi-objective optimization algorithms in Section IV. Our methods for solving the multi-objective workflow optimization problem are described in Section V. Experimental details and simulation results are presented in Section VI. Finally, we conclude the paper and present future work in Section VII.

II. Related Work

Many heuristics have been developed for scheduling DAG (Directed Acyclic Graph) based task graphs in multiprocessor systems [11][13]. Several projects have also investigated the use of heuristics for scheduling workflows on Grids. Min-Min, Max-Min and Sufferage were employed by Mandal et

al [6] to schedule bio-imaging applications. Genetic algorithms and HEFT (Heterogeneous Earliest Finish Time) have been extended by the ASKALON project [7][14] to schedule scientific applications in Grid environments. More recently, Singh et al [15] incorporated GA with Min-Min heuristics to optimize execution costs of provisioning resources for application execution.

In addition, several heuristics have been proposed to address scheduling problems based on users' specified QoS constraints such as budget and deadline. Tsiakkouri et al [10] developed scheduling approaches to adjust a schedule generated by a time optimized heuristic and a cost optimized heuristic to meet users' budget constraints respectively. GRIA (Grid Resources for Industrial Applications) [24] provides various resource allocation strategies for workflow execution based on QoS requirements. In our previous work [9], we have developed algorithms based on the genetic algorithms to minimize either execution cost or time.

The work in this paper is distinct from the related work because it simultaneously optimizes multiple objectives of workflow execution according to users QoS constraints and is capable of generating a set of alternative trade-off solutions for users' further decisions.

III. Workflow Planning Problem

Workflow execution planning is carried out prior to workflow execution. It aims to investigate users execution requirements and generate possible execution schedules. Below, we give the formulation of the execution optimization problem during the planning stage.

We model a workflow application as a DAG. Let Γ be the finite set of tasks $T_i (1 \leq i \leq n)$. Let Λ be the set of directed arcs of the form (T_i, T_j) where T_i is called a parent task of T_j, and T_j the child task of T_i. Associated with each directed arc is a dataflow in which the output of the parent is required as input data by the child. We assume that a child task cannot be executed until all of its parent tasks have been completed. Then, the workflow application can be described as a tuple $\Omega(\Gamma, \Lambda)$.

Let m be the number of services available. There is a set of services $S_i^j (1 \leq i \leq n, 1 \leq j \leq m_i, m_i \leq m)$, capable of executing the task T_i. Services have varied processing capability delivered at different prices. We denote that $time(T_i)$ is the completion time of T_i and $cost(T_i)$ the input data transmission cost and service cost for processing T_i. The execution optimization problem is to generate solution I, which maps every T_i onto a suitable S_i^j to achieve the multi-objective below:

$$\text{Minimize } Time(I) = \max_{T_i \in \Gamma} time(T_i) \quad (1)$$

$$\text{Minimize } Cost(I) = \sum_{T_i \in \Gamma} cost(T_i) \quad (2)$$

$$\text{subject to } Cost(I) < B$$
$$Time(I) < D$$

where B is the cost constraint (budget) and D is the time constraint (deadline) required by users for workflow execution.

IV. Multi-objective Optimization

We formalized the workflow planning problem as a multi-objective optimization problem [19] in which a group of conflicting objectives are simultaneously optimized. As given in Equation 1 and 2, there are two conflicting objectives: minimize execution time and minimize execution cost. In such problems, there is no single optimal solution but rather a set of potential solutions.

A. Definitions

We can define a multi-objective problem as:

$$\text{Minimize } f(x) = f_1(x), ..., f_n(x)$$

where $x \in X$ and X is a solution space. We say that the solution is said to dominate another if it is as good as the other solution and better in at least one objective. That is x^* dominates x, if and only if

$$\forall i \in [1, ..., n], f_i(x^*) \leq f_i(x) \wedge \exists j \in [1, ...n], f_j(x^*) < f_j(x)$$

Figure 1 shows solutions for the minimization problem of two conflicting objectives, $f_1(x)$ and $f_2(x)$. The solution x_3 dominates y_2, because both objective values of x_3 are lower than those of y_2. However, x_3 does not dominate x_4, since $f_2(x_3) > f_2(x_4)$. We say that x_3 and x_4 are non-dominated solutions. They are optimal in one objective but none of these two solutions is superior to the other with respect to the two objectives.

The set of non-dominated solutions form a non-dominated set P, which meets following conditions:

- Any solution in P is non-dominated to any other solution in P with respect to all objectives.
- Any solution in P dominates at least one solution not belonging to P.

For example, the set of solutions $x_1 - x_4$ is a non-dominated set for the given problem. The image of the non-dominated set is called *non-dominated front*.

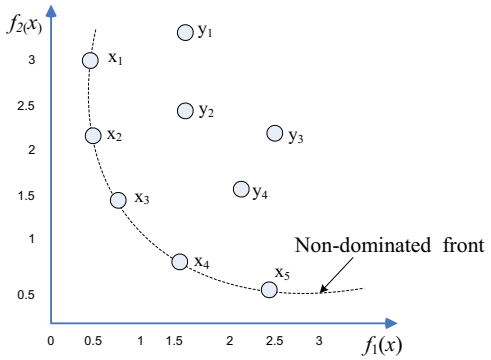


Fig. 1. Five non-dominated solutions $(x_1 - x_5)$ and four dominated solutions $(y_1 - y_4)$ for two objectives, time and cost. This is a minimization problem.

B. Multi-objective Algorithms

Scheduling of interdependent tasks in distributed heterogeneous computing environments is well known to be an NP-hard problem [17]. Many non-evolutionary based heuristics have been proposed to optimize execution time. However, with the increase in the number of parameters and objectives required to be considered, it becomes infeasible to develop a simple heuristic or use a classical method such as a linear programming approach to handle the scheduling optimization problem. Evolutionary based algorithms have been widely applied to solve multi-objective optimization problems in many application domains, such as control systems [16] and network routing [18], and are capable of simultaneously optimizing multiple objectives without combining them into a single scalar objective function.

In this paper, we focus on applying three well-known algorithms to solve the workflow execution planning problem and compare the algorithms for different workflow structures and users constraint levels. The three algorithms are: *Non-dominated Sorting Genetic Algorithm* (NSGAII) [20], *Strength Pareto Evolutionary Algorithm* (SPEA2) [21] and *Pareto Archived Evolution Strategy* (PAES) [22]. NSGAII and SPEA2 are *population-based* algorithms with different evaluation and selection schemes, while PAES is a *local search-based* algorithm.

A general high level overview [25] of NSGAII and SPEA2 algorithms is shown in Figure 2. Both NSGAII and SPEA2 are genetic algorithms-based methods and select individuals from a population for reproduction. Both crossover and mutation genetic operators are performed to generate new offspring. However, these two algorithms employ different evaluation and selection strategies.

NSGAII evaluates solutions based on the values of each objective. It ranks all solutions to form non-dominated fronts according to its values (see Figure 1). It first selects non-dominated solutions in the current population as the first level non-dominated front. It then selects non-dominated solutions in the rest of the population as the next level non-dominated front. The procedure continues until the whole population is classified into non-dominated fronts. In the selection phase, an individual's non-domination rank biases the probability of being selected for reproduction. The solutions in the first level front have highest priority, and then those in the second level and so forth. The selection process continues until it finds that the population size would be exceeded if all solutions were added to the next level front. Then, crowding sort [20] is performed on the solutions at this level and finally the solutions from the less crowded area are selected.

On the other hand, SPEA2 uses the degree to which the solution dominates other members in the population and its density estimation to evaluate solutions. The density of a solution in a population is a function of the distance to the k-th nearest solution. Unlike NSGAII, SPEA2 does not select solutions based on their non-dominated levels at each generation. It creates an external *archive* to keep selected individuals for the next generation and first copies non-dominated solutions in the current population to the archive. If the size of the archive is exceeded, the solutions in overcrowded areas are removed from the archive; otherwise, it fills the archive with dominated solutions based on their *Euclidean distance* to its nearest neighbour solution.

NSGAII and SPEA2 algorithm

```
1.  generate initial population
2.  do
3.    perform crossover on individuals
4.    perform mutation on offspring
5.    evaluate solutions
6.    select individuals to be carried into next generation
7.  while(termination condition is not satisfied)
```

Fig. 2. Generic outline of NSGAII and SPEA2.

PAES algorithm

```
1.  generate current solution
2.  create an archive
3.  do
4.    mutate current solution and generate a candidate
      solution
5.    evaluate candidate solution
6.    if the candidate solution is not dominated by
         the current solution  then
7.     compare the candidate solution with archive members
8.     update archive
9.     select a new current solution
10.   end if
11. while(termination condition is not satisfied)
```

Fig. 3. Overview of PAES.

An overview of PAES is shown in Figure 3. PAES uses local search from one current solution to generate a new candidate and compares the current solution with the candidate solution. It continues to search from the current solution if the candidate is dominated; otherwise it is compared with other archived solutions. In PAES, an archive is maintained to keep the best solutions found so far. Initially there is only one solution in the archive. As the number of generations increases, the archive is updated by adding good candidates and removing dominated archived members. In addition to comparing solutions by using the dominance criteria, density estimation is also applied.

V. Multi-objective Workflow Execution Planning

In order to extend MOEAs to solve the workflow scheduling problem, we need to define an appropriate problem representation, fitness assignment, and genetic operators. The methods we have employed are described in the following sub-sections.

A. Problem encoding

For the workflow scheduling problem, a feasible solution is required to meet the following conditions: (a) a task can only be started after all its predecessors have completed. (b) every task appears once and only once in the schedule. (c) each task must be allocated to one available time slot of a service capable of executing the task.

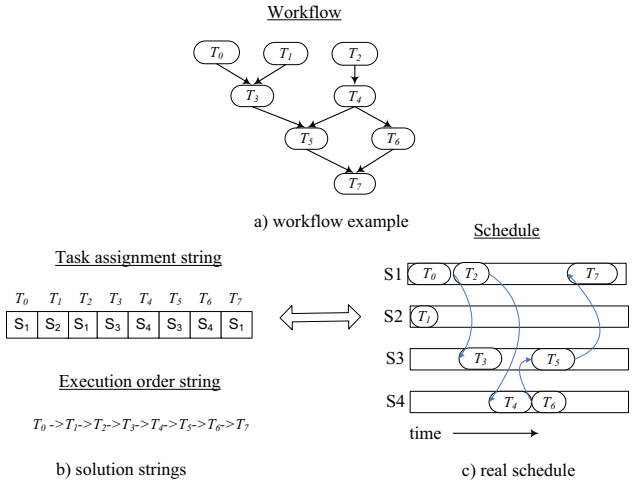


Fig. 4. Workflow representation in the search space.

Each individual in the search space represents a feasible solution to the problem. In our work, each solution is represented by two strings [12], the *task-assignment string* and the *scheduling-order string*. As shown in Figure 4, the task-assignment string encodes the allocation for each task. For example, task T_0 is assigned to service S_1. The scheduling-order string encodes the order to schedule tasks. For example, T_1 is scheduled after T_0, T_2 is scheduled after T_1 and so forth. The order in the scheduling-order string must satisfy task dependencies; that means a task should not be placed before its predecessors. However, the main reason for having the scheduling-order string is not only to code task dependencies but also the execution priorities for independent tasks which are assigned to a same service. For example, two independent tasks T_0 and T_2 are assigned on S_1, but T_0 is executed first according to the scheduling-order string.

B. Fitness function

A fitness function is used to measure the quality of the solutions according to the given optimization objectives. We separate fitness functions by objective functions and penalty functions. Objective functions are designed to encourage the algorithms to choose solutions with minimum objective values. The objective functions for solution I are defined as follows:

$$\text{Cost objective function: } f_{cost}(I) = \frac{cost(I)}{B}$$

$$\text{Time objective function: } f_{time}(I) = \frac{time(I)}{D}$$

A penalty function is developed to handle constraints. It is defined as follows:

$$P(I) = P_{budget}(I) + P_{deadline}(I)$$

where P_{budget} is the budget penalty function defined by:

$$P_{budget}(I) = \begin{cases} f_{cost}(I) & \text{if } cost(I) > B \\ 0 & \text{otherwise} \end{cases}$$

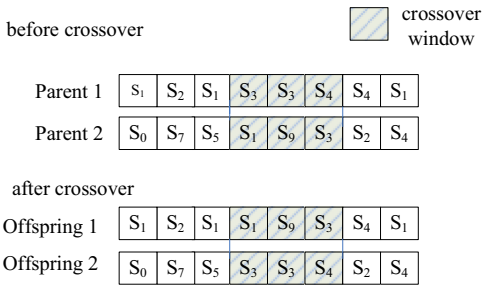


Fig. 5. Illustration of crossover operation.

and $P_{deadline}$ is the deadline penalty function defined by:

$$P_{deadline}(I) = \begin{cases} f_{time}(I) & \text{if } time(I) > D \\ 0 & \text{otherwise} \end{cases}$$

Since satisfying deadline and budget requirements is the primary goal of the scheduling scheme, the overall penalty is added to the objective functions to form the fitness functions:

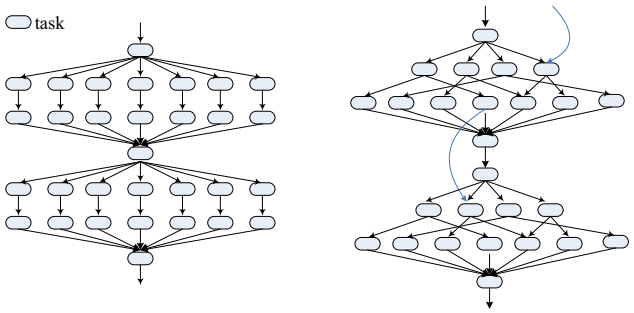
$$\text{Cost fitness function: } F_{cost}(I) = f_{cost}(I) + P(I)$$
$$\text{Time fitness function: } F_{time}(I) = f_{time}(I) + P(I)$$

C. Evolutionary operations

Evolutionary operators are used to generate new solutions based on solutions found so far. The two operators are crossover and mutation.

1) Crossover: For population-based evolutionary algorithms (e.g. NSGAII and SPEA2), crossovers are used to create new solutions by rearranging parts of the existing solutions in the current population. The idea behind the crossover is that the fittest solution may result from the combination of two of the current fittest solutions. We have implemented two-point crossover which is illustrated in Figure 5. The crossover operator is implemented as follows: (a) two parents are chosen at random in the current population. (b) two random points are selected from the task-assignment strings. (c) all tasks between these two points are chosen as successive crossover points. (d) the service allocation of all tasks within the crossover window are exchanged. After crossover, two new offspring are generated by combining task assignments taken from the two parents.

2) Mutation: For population based algorithms, mutations occasionally occur in order to allow a child to obtain features that are not possessed by either of its parents. This process helps the algorithm to explore new and possibly better genetic material than previously considered. For local search-based algorithms (e.g. PAES), mutation operations are used to generate a new solution based on the current solution. We have developed two types of mutations: *reordering mutation* and *replacing mutation* for the workflow scheduling problem. The reordering mutation aims to change the execution order of independent tasks on the same service, while the replacing mutation aims to re-allocate an alternative service to a task in a solution.



a) balanced-structure application b) unbalanced-structure application

Fig. 6. small portion of workfbw applications.

VI. Experiments

A. Workflow applications

Given that different workflow applications may have different impact on the performance of the scheduling algorithms, we have evaluated the algorithms on different workflow structures. According to many Grid workflow projects [7][8], workflow applications can be categorized into *balanced structure* and *unbalanced structure*. Figure 6 shows balanced- and unbalanced-structure applications used in our experiments. As shown in Figure 6a, the balanced-structure application consists of several parallel pipelines, which require the same types of services but process different data sets. In Figure 6b, the structure of the unbalanced-structure application is more complex. Unlike the balanced-structure application, many parallel tasks in the unbalanced structure require different types of services, and their workload and I/O data varies significantly.

B. Experimental setting

In our experiments, we have simulated 15 types of services with various price levels, each of which was supported by 10 different service providers with varied processing capability. The topology of the system is such that all the services are connected to one another, and the available network bandwidths between the services are 100Mbps, 200Mbps, 512Mbps or 1024Mbps. The processing cost and transmission cost are inversely proportional to the processing time and transmission time respectively.

The behaviors of algorithms are also observed at three constraint levels, namely relaxed constraint, medium constraint, and tight constraint. The relaxed constraint level assumes that users require relatively large deadline and budget, while tight constraint level assumes that users require low deadline and budget. The relaxed/tight deadlines and budgets of an application are determined by the maximum and minimum time/cost for the workflow execution. T_{max} is the time derived by executing the workflow at the cheapest cost C_{min}, while C_{max} is the cost achieved by executing the workflow at the shortest time T_{min}. The T_{max} and C_{min} can be generated by an cost optimization algorithm such GreedyCost [9], while T_{min} and C_{max} are generated by a time optimization algorithm such as HEFT [11]. The deadline D and budget B are defined by Equation 3 and 4 respectively.

$$D = T_{max} - k(T_{max} - T_{min}) \quad (3)$$

$$B = C_{max} - k(C_{max} - C_{min}) \quad (4)$$

The value of k was varied from 0.2 to 0.8 to generate the relaxed, medium and tight deadlines and budgets.

TABLE I
Default settings.

Parameter	Value/Type
Population size	10
Initial population	randomly generated solutions
Maximum generation(NSGAII, SPEA2)	100
Crossover probability(NSGAII, SPEA2)	0.9
Mutation probability	0.5
Archive size(SPEA2, PAES)	10
Depth(PAES)	10
Maximum iteration(PAES)	1000

The parameter settings used as the default configuration for NSGAII, SPEA2 and PAES are listed in Table I. Two versions of SPEA2 and NSGAII have been used. In the first version, the initial population was randomly generated. In the second version, we seed the population with individuals generated by simple heuristics. Here, we have implemented two heuristics: Greedy Cost-Time Distribution (TD) and Greedy Time-Cost Distribution (CD). The CD approach is aimed at minimizing execution time while meeting the budget constraint, while the TD is aimed at minimizing execution cost while meeting the deadline constraint.

C. Comparison of Different algorithms

In our first experiment, we compare three algorithms, NSGAII, SPEA2 and PAES, using relaxed, medium and tight constraints for unbalanced-and balance-structure applications. NSGAII and SPEA2 were selected with two different initial populations. Two of the members in the initial population were comprised of the solutions produced by two deadline or budget constrained minimization heuristics, TD and CD, which are described in Section VI-B, together with other randomly generated solutions. We denote the results returned by NSGAII and SPEA2 with this type of initial population as NSGAII* and SPEA2* respectively. The other initial population contains only randomly generated solutions and corresponding results returned by NSGAII and SPEA2 are denoted as NSGAII and SPEA2 respectively.

In order to compare the performance of alternative workflow multi-objective scheduling algorithms, we need to examine the extent of minimization of the obtained non-dominated solutions produced by each algorithm for each objective and the spread of their solutions. Figure 7 show the non-dominated solutions obtained at the end of simulation trial (average over 10 runs) for the unbalanced-structure application. The performance appears to be dramatically different when the TD and CD heuristics have been used to initialize the population. Compared with SPEA2 and NSGAII, SPEA2* and NSGAII* guarantee the extreme solutions and have a better spread distribution of solutions within the constraints, whereas the

solutions obtained by SPEA2 and NSGAII dominate a small subset of solutions produced by SPEA2* and NSGAII*. However, as the constraints are tightened, SPEA2* and NSGAII* significantly outperforms SPEA2 and NSGAII in terms of the minimization and distribution of solutions. As shown in Figure 7c, the solutions produced by SPEA2 and NSGAII cannot meet the deadline and budget constraints. This shows that the performance of population-based algorithms can be significantly affected when we initialize the population with solutions generated by TD and CD.

In order to present a comprehensive comparison of the overall quality of these alternative approaches, we have run each algorithm for two different workflow structures at relaxed, medium and tight constraint levels respectively. The experiment for each scenario was repeated 30 times (the average running time of each experiment on a machine with Intel CPU 2.00GHz was 4 mins). We have constructed a reference set, R, by merging all of the archival non-dominated solutions found by each of the algorithms for a given workflow structure and constraint level across 30 runs. We then use the hypervolume difference indicator I_H^- to measure the differences between non-dominated fronts generated by the algorithms and the reference set R. I_H^- measures the portion of the objective space that is dominated by R [23]. The lower the value of I_H^-, the better the algorithms perform.

The box plots in Figure 8 and 9 show statistical significance difference between algorithms for balanced- and unbalanced-structure workflow respectively (p-value < 0.05). We can observe that NSGAII and SPEA2 perform similar on both applications, whereas PAES performs differently. For the balanced-structure application, as shown in Figure 8 PAES performs worst at relaxed and medium constraint while it performs very similar to SPEA2 and NSGAII at tight constraint. However, it outperforms SPEA2 and NSGAII for the unbalanced-structure application as we can see from Figure 9. Therefore, it can be said that the local search-based algorithm is more efficient for the unbalanced-structure application whereas the population-based algorithms is better for the balanced-structure application.

However, a population-based algorithm can be improved by employing solutions optimizing each objective separately as initial individuals. The statistical analysis in Figure 8 and 9 shows that SPEA2* is significantly better than SPEA2 and outperforms PAES. Even though the behaviors of SPEA2 and NSGAII are similar, NSGAII* does not perform as well as SPEA2*. It even performs worse than NSGAII on the unbalanced-structure application at relaxed constraint (see Figure 9a).

The major difference between the SPEA2 algorithm and the NSGAII algorithm is their selection strategies. Different selection strategies mean that NSGAII and SPEA2 perform differently. As described in Section IV-B, SPEA2 firstly selects non-dominated solutions in the current population, and then select other solutions which are in less overcrowded area. On the other hand, NSGAII not only selects the non-dominated solutions in the population, but also ranks the rest of the solutions into non-dominated levels and selects solutions belong to higher non-dominated levels. Such an elitist selection strategy could result in less diverse elements contained by individuals and thus leading to premature convergence, since it easily chooses the individuals which inherit the parts from employed initial individuals. However, as the tight degree of the constraint increase, the search space, which contains solutions satisfying constraints decreases. Therefore, the difference between the SPEA2* and NSGAII* decreases as the constraints become tight. As shown in Figure 8c and Figure 9c, the difference between NSGAII* and SPEA2* is not significant. These results show that with incorporating TD and CD, SPEA2 can be significantly improved and perform best in all the cases, while the performance of NSGAII gets worse for the unbalanced-structure workflow with relaxed and medium constraints. As shown in Figure 8 and 9, SPEA2* can perform better than NSGAII* for both workflow applications.

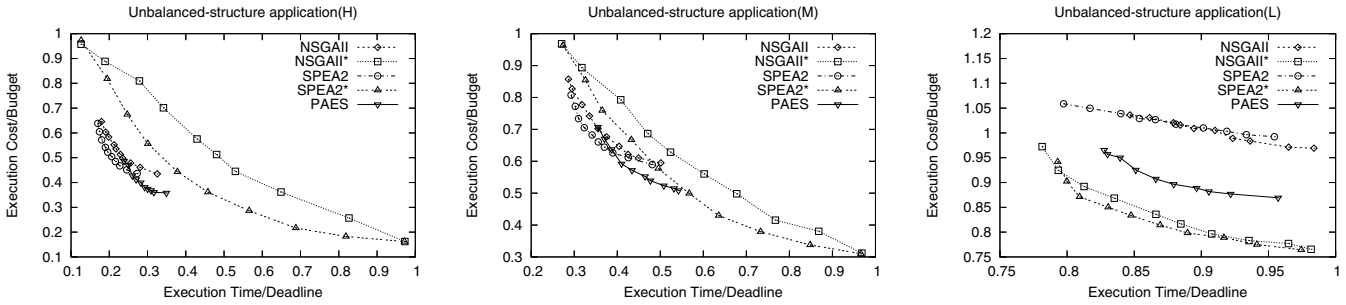
D. Effect of changing crossover type

We also compare the crossover operator (denoted as two-point crossover) proposed in Section V-C.1 with another crossover type called one-point crossover [12]. One-point crossover divides the task-assignment string into a top part and bottom part and swaps task assignments of bottom parts of parents. As shown in Figure 10, the results generated by two crossover types are similar for the balanced-structure application. However, the two-point crossover can achieve better performance for the unbalanced-structure application. This means that two-point crossover can encourage the evolutionary algorithm to search better solutions more efficiently.

VII. Conclusion

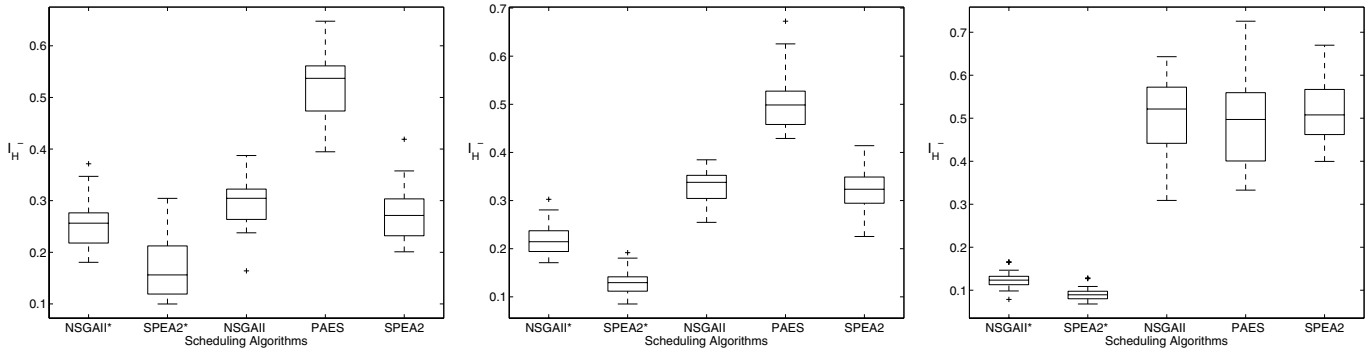
Existing workflow scheduling algorithms only attempt to minimize either execution time or execution cost. However, more scheduling objectives are required to be considered while scheduling workflows on utility Grids. In this paper, we have proposed a workflow execution planning approach, which optimize multiple objectives. The planner can generate a set of widespread alternative solutions if the optimization objectives are conflicted. Providing these alternative solutions can offer more flexibility to users to estimate their preferences and choose a desired workflow schedule based on their QoS requirements.

We have applied Multi-objective Evolutionary Algorithms (MOEAs) for the workflow execution planning problem. Our goal was to simultaneously minimize two conflicting objectives-execution time and execution price while meeting users' maximum time constraint (deadline) and price constraint (budget). Corresponding fitness functions, which incorporate minimization objectives and penalty functions for the constraints, have been developed. We have also compared two population-based MOEAs, NSGAII and SPEA2, and one local search-based MOEAs, PAES. A statistical analysis of the result has been presented to show the quality of each algorithm for various workflow structures and constraint levels. We have also proposed a method which incorporates single objective



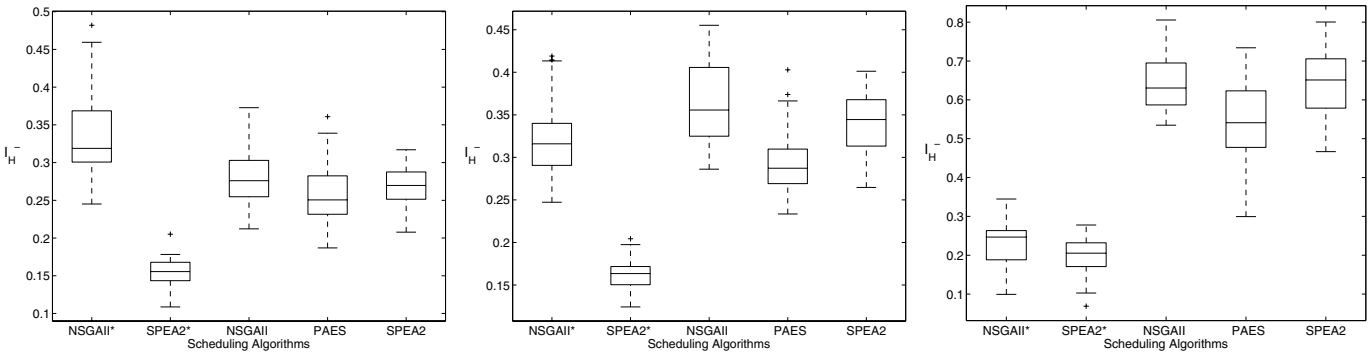
(a) relaxed constraint (k=0.2) (b) medium constraint (k=0.5) (c) tight constraint (k=0.8)

Fig. 7. Obtained non-dominated solutions for the unbalanced-structure application on different constraint levels.



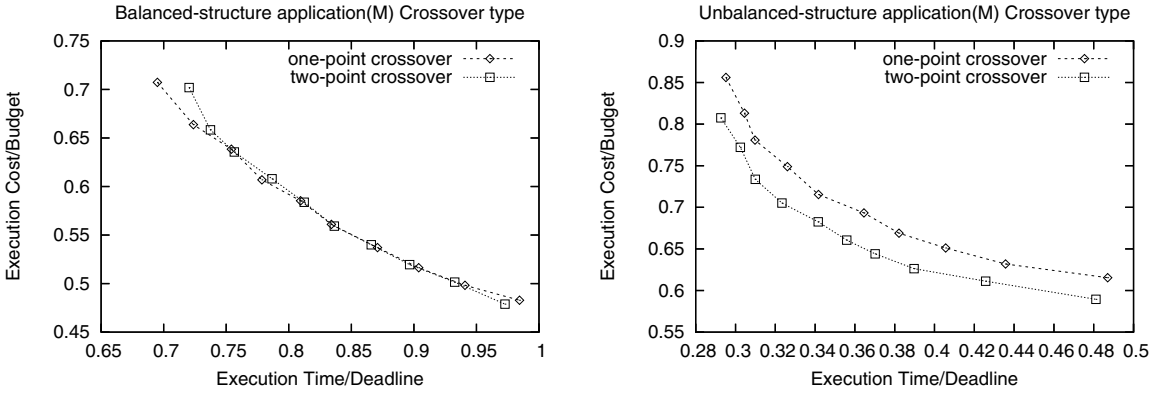
(a) relaxed constraint (b) medium constraint (c) tight constraint

Fig. 8. Box plot of I_H^- indicator values for the balanced-structure application on different constraint levels.



(a) relaxed constraint (b) medium constraint (c) tight constraint

Fig. 9. Box plot of I_H^- indicator values for the unbalanced-structure application on different constraint levels.



(a) balanced-structure

(b) unbalanced-structure

Fig. 10. Performances of SPEA2 with different crossover types.

optimization heuristics and population-based algorithms. The simulation results show that it can significantly improve the performance of SPEA2 and find a range of compromise solutions within a short computational time.

In future work, we will evaluat scheduling approaches in terms of pricing models and Grid sizes. We will also enhance the work by providing run-time rescheduling approaches to support adaptive execution environments.

ACKNOWLEDGMENT

We would like to thank Robert Stewart for providing scripts for statistical analysis. We also want to thank Srikumar Venugopal, Hussein Gibbins and Chee Shin Yeo for their comments on this paper. This work is partially supported through Australian Research Council (ARC) Discovery Project grant.

REFERENCES

[1] I. Foster et al., The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Technique Report, Gridbus Project,2002.

[2] T. Eilam et al.,Using a utility computing framework to develop utility systems, *IBM System Journal*,43:97-120,2004.

[3] E. Deelman et al., Mapping Abstract Complex Workfbws onto Grid Environments, *Journal of Grid Computing*, 1:25-39, 2003.

[4] T. Oinn et al., Taverna: a tool for the composition and enactment of bioinformatics workfbws. *Bioinformatics*, 20(17):3045-3054, Oxford University Press, London, UK, 2004.

[5] G. V. Laszewski, Java CoG Kit Workfbw Concepts for Scientifi c Experiments, Argonne National Laboratory, Argonne, IL, USA Technique Report, 2005.

[6] A. Mandal et al., Scheduling Strategies for Mapping Application Workfbws onto the Grid, in *IEEE International Symposium on High Performance Distributed Computing* (HPDC 2005), Research Triangle Park, NC, USA, 2005.

[7] M. Wieczorek et al., Scheduling of scientifi c workfbws in the ASKALON grid environment, *SIGMOD Rec.*,34:56-62, 2005.

[8] J. Blythe et al., Task scheduling strategies for workfbw-based applications in grids, in *Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid* (CCGrid'05), 2005.

[9] J. Yu and R. Buyya, Scheduling Scientifi c Workfbw Applications with Deadline and Budget Constraints using Genetic Algorithms, *Scientific Programming*,14:217-230, 2006.

[10] E. Tsiakkouri et al., Scheduling Workfbws with Budget Constraints, in *CoreGRID Integration Workshop* Pisa, Italy, 2005.

[11] T. Haluk et al., Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing, *IEEE Transactions on Parallel and Distributed Systems.*, 13:260-274, 2002.

[12] W. Lee et al., Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach, *J. Parallel Distributed Computing.*,47:8-22, 1997.

[13] E. S. H. Hou et al., A Genetic Algorithm for Multiprocessor Scheduling, *IEEE Transactions on Parallel Distributed Systems.*,5:113-120, 1994.

[14] R. Prodan and T. Fahringer, Dynamic scheduling of scientifi c workfbw applications on the grid: a case study,in *Proceedings of the 2005 ACM symposium on Applied computing*, Santa Fe, New Mexico, 2005.

[15] S. Gurmeet et al., Application-Level Resource Provisioning on the Grid, in *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*, 2006.

[16] P. J. Fleming and R. C. Purshouse, Evolutionary Algorithms in Control Systems Engineering: a Survey, *Control Engineering Practice*, 10:1223-1241, 2002.

[17] J. D. Ullman, NP-complete Scheduling Problems, *Journal of Computer and System Sciences*, 10:384-393, 1975.

[18] A. Roy and S. K. Das, Optimizing QoS-Based Multicast Routing in Wireless Networks: A Multi-Objective Genetic Algorithmic Approach, in *Second IFIP-TC6 Networking Conference*(Networking 2002), Pisa, Italy, 2002.

[19] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. England, Wiley and Sons, 2001.

[20] K. Deb et al., A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II, *Parallel Problems Solving from Nature VI*, pp.849-858, 2000.

[21] E. Zitzler et al., SPEA2: Improving the strength Pareto evolutionary algorithm,Technique Report, Swiss Federal Institute of Technology 2001.

[22] J. D. Knowles and D. W. Corne, The Pareto Archive Evolution Strategy: A New Baseline Algorithm for Multi-Objective Optimization, *The congress on Evolutionary Computation*, pp. 98-105, 1999.

[23] E. Zitzler and L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach, *IEEE Transactions on Evolutionary Computation*, 3:257-271, 1999.

[24] M. Ghanem et al., Grid-enabled Workfbws for Industrial Product Design, *Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*, Amsterdam, The Netherlands, 2006.

[25] PISA, A Platform and Programming Language Independent Interface for Search Algorithms, http://www.tik,ee,ethz.ch/sop/pisa/.