CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2009)
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.1417

Multiobjective differential evolution for scheduling workflow applications on global Grids

A. K. M. Khaled Ahsan Talukder[*, †], Michael Kirley and Rajkumar Buyya

Department of Computer Science and Software Engineering, The University of Melbourne, Parkville, Victroia 3010, Australia

SUMMARY

Most algorithms developed for scheduling applications on global Grids focus on a single Quality of Service (QoS) parameter such as execution time, cost or total data transmission time. However, if we consider more than one QoS parameter (e.g. execution cost and time, which may be in conflict) then the problem becomes more challenging. To handle such scenarios, it is convenient to use heuristics rather than a deterministic algorithm. In this paper, we have proposed a workflow execution planning approach using Multiobjective Differential Evolution (MODE). Our goal was to generate a set of trade-off schedules according to two user specified QoS requirements (time and cost), which will offer more flexibility to users when estimating their QoS requirements. We have compared our results with a well-known baseline algorithm 'Pareto-archived Evolutionary Strategy (PAES)'. Simulation results show that the modified MODE is able to find significantly better spread of compromise solutions compared with that of PAES. Copyright © 2009 John Wiley & Sons, Ltd.

Received 7 November 2008; Accepted 6 January 2009

KEY WORDS: Grid scheduling; multiobjective optimization; multiobjective differential evolution; QoS; Pareto-front; Grid workflow scheduling

[*]Correspondence to: A. K. M. Khaled Ahsan Talukder, Grid and Distributed Systems (GRIDS) Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, ICT Building, 111 Barry Street, Carlton, Victroia 3010, Australia.
[†]E-mail: akmkat@csse.unimelb.edu.au

Contract/grant sponsor: Australian Research Council Discovery Grant; contract/grant number: DP0664674

1. INTRODUCTION

Grid Computing aims to allow unified access to data, computing power, sensors and others resources through a single virtual laboratory [1]. Technologies should provide services, protocols and software needed for the flexible and controlled sharing of resources. However, such infrastructure has to be competitive in terms of costs to the final users. Usually this corresponds to identifying the optimal use of the resources.

There are two approaches typically used to solve this optimization problem. The first is based on a distributed resources discovery and allocation system. The second is based on a central repository of resources and resources requests. The first is suitable for small jobs that can easily be accepted by the computing Grid. The second is suitable for large periodical jobs that are scheduled in advance. In this case, the centralized scheduler is usually referred to as 'metascheduler', because of its position on top of the local schedulers. It should optimize the allocation of a job allowing the execution on the 'fittest' set of resources. In most cases, such as in scientific and enterprise domains, a typical application is constructed as workflows. Here, the metascheduler will deploy the workflow on the Grid considering the current availability of resources, time constraints and user specified quality of service (QoS).

A number of Grid workflow management systems [2,3] have been developed to facilitate the composition and execution of workflow applications over distributed resources. Many heuristics [4–6] 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. In this paper, we introduce a model that can optimize conflicting objectives. However, in many Differential Evolution (DE)-based Job-Shop/Flow-Shop scheduling, the algorithm was implemented by mapping Job/Machine sequence to real numbers. As this approach is not feasible in the case of Grid scheduling, we have to deal with exact scheduling sequences and thus our approach does not need to map the resource/task sequence to real values as described in [7,8].

2. MOTIVATION AND RELATED WORK

The application of nature's heuristics for scheduling of applications has been explored by various researches [9–12]. There are numerous studies reporting work done on scheduling Directed Acyclic Graph (DAG)-based task graphs in multiprocessor systems [5]. Genetic algorithms and Heterogeneous Earliest Finish Time have been extended by the ASKALON project [13,14] to schedule scientific applications in Grid environments. Recently, in [15], a different base-line algorithm for Multiobjective Optimization was tested on different workflow models.

Evolutionary techniques are now a widely used approach for tackling complex multiobjective optimization problems. We find numerous examples of them in the literature. For example, in [16] the authors proposed a local-search-based enhancements for evolutionary computation. Similar approaches can be found in [17–20]. Evolutionary and genetic-based optimization techniques are also found in other Grid-oriented applications (especially in Data-Grid applications), for review, readers are encouraged to refer to [21,22]. In a similar study [23], the proposed model was compared with Max–Min and Min–Min heuristics described in [4]. Although most of the techniques in this domain only considers single objective problems, however we have also found some multiobjective instances, such as in [24]. For more elaborate survey and literature review, readers are referred to [25,26].

As the main objective of this research is to devise a novel approach for *DE*-based technique, we have also reviewed some examples in [7,27]. However, DE-based techniques are specially suitable for numerical optimization problems; hence, most of the models in the literature are implemented in such a way that they can cope with the basic working principles of DE. For this purpose, those models use the basic formulations as stated below:

- As in the case of scheduling problem, each gene in a chromosome should represent corresponding job or machine/resource sequence.
- DE is mainly used for numerical optimization problems, hence, the combination of job/resource sequence needs to be converted to a range of real numbers.
- Hence, in most cases, there is a mapping procedure (as well as a inverse mapping procedure) from job/resource sequence to real numbers.

The fact is that this kind of forward and inverse mapping may lead to creation of infeasible solutions. Moreover, to avoid this discrepancy, we need to implement a good repairing algorithm that eventually leads to an extra computational cost to the overall algorithm. The work in this paper is distinct from the related work because we do not adopt the same scheme (mapping of job/resource sequence to real numbers). More specifically, we apply the relevant DE operations on exact schedule rather than on sequence of real numbers. Hence our model can simultaneously optimize multiple objectives of workflow execution according to users' QoS without the unwanted creation of infeasible solutions.

3. PROBLEM OVERVIEW

We formalized the workflow planning problem as a multiobjective optimization problem (MOP) in which a group of conflicting objectives are simultaneously optimized. As given in Equations (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. We can define a multiobjective problem as

$$\textit{minimize}\ f(x) = f_1(x), f_2(x), \ldots, f_k(x) \tag{1}$$

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

$$\forall i \in [1, 2, \ldots, k], \quad f_i(x^*) \leq f_i(x) \wedge \exists j \in [1, 2, \ldots, k], \quad f_j(x^*) < f_j(x) \tag{2}$$

The resource scheduling problem for the Grid consists of arranging the pairs of jobs and resources given certain constraints. The objective is to minimize the multi-dimensional QoS metrics. We regard completion time of jobs and total execution cost of jobs as multi-dimensional QoS metrics.

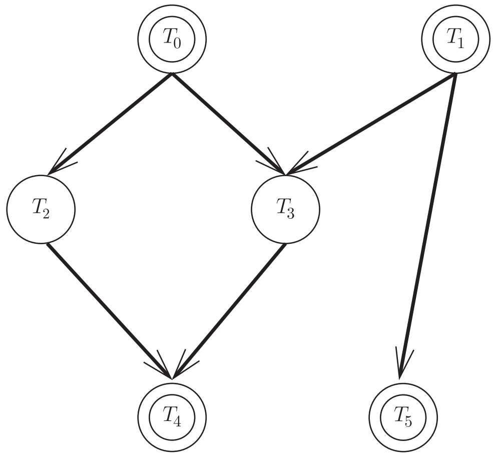


Figure 1. A simple DAG.

To address the problem, we have to start with the following assumptions:

- There are dependent relationships between jobs, and relationships can be denoted by DAG as in Figure 1.
- Jobs come in batch mode.
- Resources cannot be occupied by other jobs when a job is running on them.

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 data flow 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 $\Omega(\Gamma, \Lambda)$.

Given a set of jobs $T = \{T_i\}$, $i = 1, 2, \ldots, n$ and a set of services $S = \{S_i\}$, $j = 1, 2, \ldots, m$, under the constraint that there are dependent relationships among jobs. The execution optimization problem is to generate a solution $I = (T_i, f_j(T_i))$ where $f_j(T_i)$ is a mapping function that assigns T_i onto service S_j. We also denote that $time(T_i, f_j(T_i))$ is the completion time of I and $cost(T_i, f_j(T_i))$ is the input data transmission cost and service cost for processing I. The goal is described as follows:

$$\textit{minimize} \quad F = (F_1, F_2) \tag{3}$$

$$F_1 = \max_{T_i \in \Gamma, f_j(T_i) \in \Lambda} time(T_i, f_j(T_i)) \tag{4}$$

$$F_2 = \sum_{T_i \in \Gamma, f_j(T_i) \in \Lambda} cost(T_i, f_j(T_i)) \tag{5}$$

$$\text{subject to} \quad F_1 < B \text{ and } F_2 < D$$

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

4. MULTIOBJECTIVE DIFFERENTIAL EVOLUTION

The idea of multiobjective differential evolution (MODE) was first introduced in [28]. In DE [29], new candidate solutions are created by combining the parent individual and several other individuals of the same population. A candidate replaces the parent only if it has a better fitness value. This is a rather greedy selection scheme that often outperforms traditional Evolutionary Algorithms. Actually, there are many variants of candidate creation procedures in DE. We use the DE scheme **DE/rand/1/bin** described in Algorithm 1 (more details on this and other DE schemes can be found in [29]). Besides preserving a uniformly spread front of non-dominated solutions, which is a challenging task for any Multiobjective Evolutionary Algorithm (MOEA), we have to deal with another question, that is, when to replace the parent with the candidate solution. In single-objective optimization, the decision is easy: the candidate replaces the parent only when the candidate is better than the parent.

```
Algorithm 1 CreateCandidate(Parent $P_i\langle mat, ss\rangle$)
  Pick three random individual $P_{i1}$, $P_{i2}$ and $P_{i3}$ where $(i1 \neq i2 \neq i3)$
  Calculate Candidate C as, $C = P_{i1} + F \cdot (P_{i2} - P_{i3})$
  Modify the Candidate by binary crossover with the Parent
  return Candidate C
```

In MOPs, on the other hand, the decision is not so straightforward. We could use the concept of dominance (the candidate replaces the parent only if it dominates it), but this would make the greedy selection scheme of DE even greedier. Therefore, MODE applies the following principle (see Algorithm 2):

1. The candidate replaces the parent if it dominates it.
2. If the parent dominates the candidate, the candidate is discarded.
3. Otherwise (when the candidate and parent are non-dominated with regard to each other).
4. The candidate is added to the population. This step is repeated until *popSize* number of candidates are created.

After that, we get a population of the size between *popSize* and $2 \cdot$ *popSize*. If the population grows, we have to truncate it to prepare it for the next step of the algorithm.

4.1. The model

In order to extend MOEAs to solve the workflow scheduling problem, we need to define an appropriate problem representation, fitness assignment and genetic operators. We have extended our ideas from [5,30]. The methods we have employed are described in the following subsections.

Algorithm 2 MODE

```
Define population $P$ with $popSize$ number of individual.
$P = \{P_1, P_2, \ldots, P_{popSize}\}$
Evaluate the initial population $P$ of random individuals.
while stopping criterion not met do
  for all $i$ such that $0 \leq i \leq popSize$ do
    $C$=CreateCandidate($P_i$)
    Evaluate Candidate
    if Candidate $C$ dominates Parent $P_i$ then
      Candidate replaces the Parent
    else if Parent $P_i$ dominates Candidate $C$ then
      Parent replaces the Candidate
    else
      Add Candidate $C$ to the population
      $popSize \Leftarrow popSize + 1$
    end if
  end for
  If the population exceeds $popSize$, truncate it.
  Randomly enumerate individuals in $P$
end while
```

4.1.1. Encoding

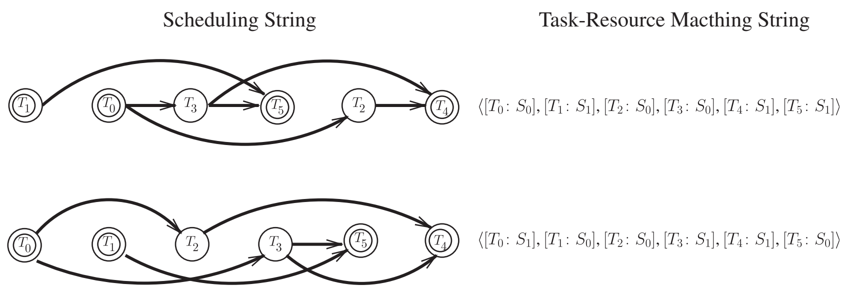
In this approach, each chromosome consists of two parts: the matching string and the scheduling string. Let mat be the matching string, which is a vector of length $|T|$, such that

$$mat(i) = j \quad \text{where } 0 \leq i < |T| \quad \text{and} \quad 0 \leq j < |S|$$

i.e. subtask T_i is assigned to service S_j.

In terms of string representation it is $T_i : S_j$ The scheduling string is a topological sort of the DAG, i.e. a total ordering of the nodes (subtasks) in the DAG that obeys the precedence constraints. Define ss to be the scheduling string, which is a vector of length $|T|$, such that $ss(k) = i$, where $0 \leq i, k < |T|$, and each T_i appears only once in the vector, i.e. subtask T_i is the kth subtask in the scheduling string.

Because it is a topological sort, if $ss(k)$ is a consumer of a global data item produced by $ss(j)$, then $j < k$. The scheduling string gives an ordering of the subtasks that is used by the evaluation step. Then in this approach, a chromosome is represented by a two-tuple $< mat, ss >$. Thus, a chromosome represents the subtask-to-service assignments (matching) and the execution ordering of the subtasks assigned to the same service. The scheduling of the *global data item transfers* and the *relative ordering of subtasks assigned to different services* is determined by the evaluation step. Figure 2 illustrates two different chromosomes for the DAG in Figure 1, for $|T| = 6$ and $|S| = 3$.


Scheduling String
Task-Resource Macthing String
T_1 T_0 T_3 T_5 T_2 T_4
T_0 T_1 T_2 T_3 T_5 T_4

$\langle [T_0 : S_0], [T_1 : S_1], [T_2 : S_0], [T_3 : S_0], [T_4 : S_1], [T_5 : S_1] \rangle$

$\langle [T_0 : S_1], [T_1 : S_0], [T_2 : S_0], [T_3 : S_1], [T_4 : S_1], [T_5 : S_0] \rangle$

Figure 2. Two sample chromosomes from the DAG in Figure 1.

4.1.2. *Initial population*

In the initial population, a predefined number of solutions (chromosomes) are generated. When generating a chromosome, a new matching string mat is obtained by randomly assigning each subtask to a machine. To form a scheduling string ss, the DAG is first topologically sorted to form a basis scheduling string. For each chromosome in the initial population, this basic string is mutated a random number of times (between one and the number of subtasks) using the mutation operator (described in Section 5.5) to generate the ss vector (which is a valid topological sort of the given DAG). Furthermore, it is common to incorporate solutions from some non-evolutionary heuristics into the initial population, which may reduce the time needed for finding a satisfactory solution. Since every time we consider a topological sort of the DAG, there is a non-zero probability that a chromosome will represent an infeasible workflow. Thus, this representation conforms to any possible (valid) solution, even after the crossover and the mutation operations are carried out. (The crossover and the mutation will be discussed in the later sections).

4.1.3. *Fitness measure*

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:

Cost objective function:

$$f_{cost}(I) = cost(I)/B \tag{6}$$

Time objective function:

$$f_{time}(I) = time(I)/D \tag{7}$$

A penalty function $P(I)$ is developed to handle constraints. It is defined as follows:

$$P(I) = P_{budget}(I) + P_{deadline}(I) \tag{8}$$

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

$$P_{budget} = \begin{cases} f_{cost}(I) & \text{if } cost(I) > B \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

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

$$P_{deadline} = \begin{cases} f_{time}(I) & \text{if } time(I) > D \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

As 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:

Cost fitness function:

$$F_{cost}(I) = f_{cost}(I) + P(I) \tag{11}$$

Time fitness function:

$$F_{time}(I) = f_{time}(I) + P(I) \tag{12}$$

4.1.4. Genetic operators

The crossover operator for the scheduling strings randomly chooses pairs of the scheduling strings. For each pair, it randomly generates a cutoff point, which divides the scheduling strings of the pair into top and bottom parts. Then, the subtasks in each bottom part are reordered. In this way, in each crossover, we can create a valid schedule. Figure 3 demonstrates such a scheduling string crossover process. On the other hand, for the crossover operator for the matching strings, (see right side image of Figure 4) first we randomly choose pairs of sequence from the matching string. For each pair, we randomly select a cutoff point to divide both matching strings into two parts. Then the machine assignments of the bottom parts are exchanged. However, in our approach the crossover operator will be applied to both matching string and scheduling string $\langle mat, ss \rangle$.

However, the mutation will only be applied to scheduling string $\langle ss \rangle$. This operator works by randomly choosing scheduling strings and it randomly selects a *victim* subtask (we will consider

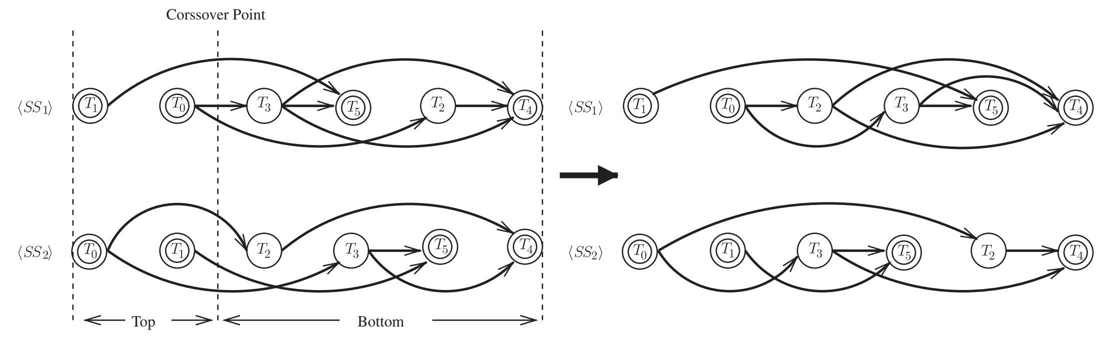


Figure 3. Scheduling string crossover.

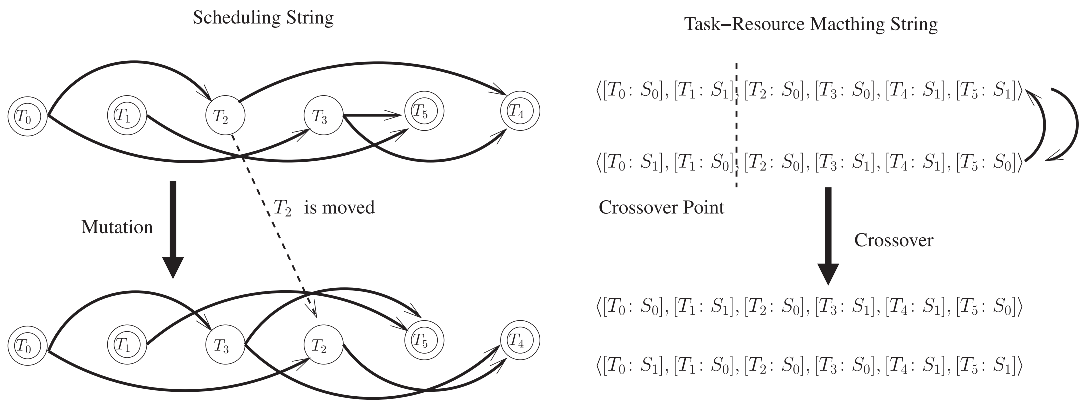

Scheduling String
Task–Resource Macthing String
$\langle[T_0: S_0], [T_1: S_1], [T_2: S_0], [T_3: S_0], [T_4: S_1], [T_5: S_1]\rangle$
$\langle[T_0: S_1], [T_1: S_0], [T_2: S_0], [T_3: S_1], [T_4: S_1], [T_5: S_0]\rangle$
Mutation
T_2 is moved
Crossover Point
Crossover
$\langle[T_0: S_0], [T_1: S_1], [T_2: S_0], [T_3: S_1], [T_4: S_1], [T_5: S_0]\rangle$
$\langle[T_0: S_1], [T_1: S_0], [T_2: S_0], [T_3: S_0], [T_4: S_1], [T_5: S_1]\rangle$

Figure 4. Scheduling string mutation and matching string crossover.

it as a 'victim' subtask). The valid range of the victim subtask is the set of the positions in the scheduling string at which this (victim subtask) can be placed without violating any data dependency constraints. Figure 4 shows an example of this mutation process. However, in the case of matching string *mat*, there will be no mutation, rather it will be subjected to the candidate creation of MODE (which will be discussed in the following sections). Hence, though our model only takes scheduling string for normal genetic operation, only the matching string will go through MODE operations.

4.2. Creation of candidate solution

As described in Section 4, the creation of candidate solution is done within step 2 of Algorithm 1 where candidate C is created as follows:

$$C = P_{i1} + F \cdot (P_{i2} - P_{i3}) \tag{13}$$

The matching string is an array of numbers or characters (each of the characters defines a service to be used by a specific task).

Rationally, it seems quite confusing to measure the quantity $P_{i2}-P_{i3}$ from two matching strings of P_{i2} and P_{i3}. Let us suppose that the matching strings of P_{i2} and P_{i3} are $P_{i2}\langle mat\rangle = \{1, 2, 3, 4, 5, 6\}$ and $P_{i3}\langle mat\rangle = \{2, 5, 3, 1, 4, 6\}$, respectively. When we compare two ordinal vectors, we will consider the first one as a 'reference' and the second one as the 'target'. In [31], the 'reference'-vector is denoted as 'pattern-vector' and the 'target'-vector is denoted as 'disorder-vector'. Similarly, we considered $P_{i2}\langle mat\rangle = \{1, 2, 3, 4, 5, 6\}$ as pattern-vector and $P_{i3}\langle mat\rangle = \{2, 5, 3, 1, 4, 6\}$ as disorder-vector. Our goal is to measure the similarity between strings. In our experiment we have used the Ulam distance [31] to measure the quantity $(P_{i2} - P_{i3})$. Since it measures the disorder of ordinal variables by counting the minimum number of 'Delete–Shift–Insert' operations. Hence, in the above case, the Ulam distance will be 2, since in $P_{i3}\langle mat\rangle$, 1 should be moved to place between 2 and 5, and again 5 should be moved to a place between 4 and 6 (Total 2

‘Delete–Shift–Insert’ operations). For simplicity, we have chosen $F=1$. However, in the case of general DE, each of the variables is real numbers and the fitness is measured in terms of mathematical operations on those real numbers. But in our case, each of these variables (characters/numbers) refers to different services, hence, we cannot map these numbers/characters to real numbers and apply mathematical operations as in [7,8]. Consequently, in place of adding the difference with each value of P_{i1}, we mutate $P_{i1}\langle mat\rangle$. I.e., we mutated D number of genes in P_{i1} where D is the Ulam distance measure described as above. The process is done in Algorithm 3, step 3. As result of this approach, we have just replaced the operation in Equation (13) with Algorithm 1. We can now devise the overall MODE from Algorithms 1 and 2. The modified MODE is illustrated in Algorithm 4.

Algorithm 3 ModifiedCreate(Parent $P_i\langle mat, ss\rangle$)

```
  Pick three random individual $P_{i1}$, $P_{i2}$ and $P_{i3}$ where $(i1 \neq i2 \neq i3)$
  $D = Ulam\ Distance(P_{i2}\langle mat\rangle P_{i3}\langle mat\rangle)$
  Candidate,$C\langle mat\rangle \Leftarrow mutate(P_{i1}\langle mat\rangle, D)$
  Candidate,$C\langle ss\rangle \Leftarrow Mutatate_{MatchingString}(P_{i1}\langle ss\rangle)$
  $C\langle ss\rangle \Leftarrow Crossover_{SchedulingString}(C\langle ss\rangle, P_i\langle ss\rangle)$
  $C\langle mat\rangle \Leftarrow Crossover_{MatchingString}(C\langle mat\rangle, P_i\langle mat\rangle)$
  **return** $C\langle mat, ss\rangle$
```

Algorithm 4 ModifiedMODE

```
  Define population $P$ with $popSize$ number of individual.
  $P = \{P_1\langle mat, ss\rangle, P_2\langle mat, ss\rangle, \ldots, P_{popSize}\langle mat, ss\rangle\}$
  Evaluate the initial population $P$ of random individuals.
  **while** stopping criterion not met **do**
    **for all** $i$ such that $0 \leq i \leq popSize$ **do**
      $C\langle mat, ss\rangle = ModifiedCreate(P_i\langle mat, ss\rangle)$
      Evaluate Candidate
      **if** Candidate $C\langle mat, ss\rangle$ dominates Parent $P_i\langle mat, ss\rangle$ **then**
        Candidate replaces the Parent
      **else if** Parent $P_i\langle mat, ss\rangle$ dominates Candidate $C\langle mat, ss\rangle$ **then**
        Parent replaces the Candidate
      **else**
        Add Candidate $C\langle mat, ss\rangle$ to the population
        $popSize \Leftarrow popSize + 1$
      **end if**
    **end for**
    If the population exceeds $popSize$, truncate it.
    Randomly enumerate individuals in $P$
  **end while**
```

5. EXPERIMENTS

In our experiment, we have implemented the balanced workflow described in [32,33]. The workflows were designed to implement some specific parallel numerical computation problems such as Parallel Gauss–Jordan Algorithm to solve systems of Equations [32], Parallel LU decompositions [32] and Discrete Laplace Transformation [33]. The example workflow models are given in Table I.

The resources were 10 computing entities connected with each other with randomly assigned price levels. The parameter setting for the Pareto Archived Evolutionary Strategy (PAES) was taken from the original work [34] and our model also has the same settings as PAES. We have run each of the algorithms for 100 generations. The distance measure that we have used for our modified MODE was Ulam distance since it depends on the minimum number of 'Delete–Shift–Insert' operations (as described in Section 4.2) and these operations best reflect the difference of resource arrangements in two schedules. In order to compare the performance of alternative workflow multiobjective 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 5(a), (b) and (c) (left side) shows the non-dominated solutions obtained at the end of simulation trial (average over 30 runs) for the workflow structure discussed in [32,33]. For all problems, the Pareto optimal front obtained by our model is better than those found by PAES.

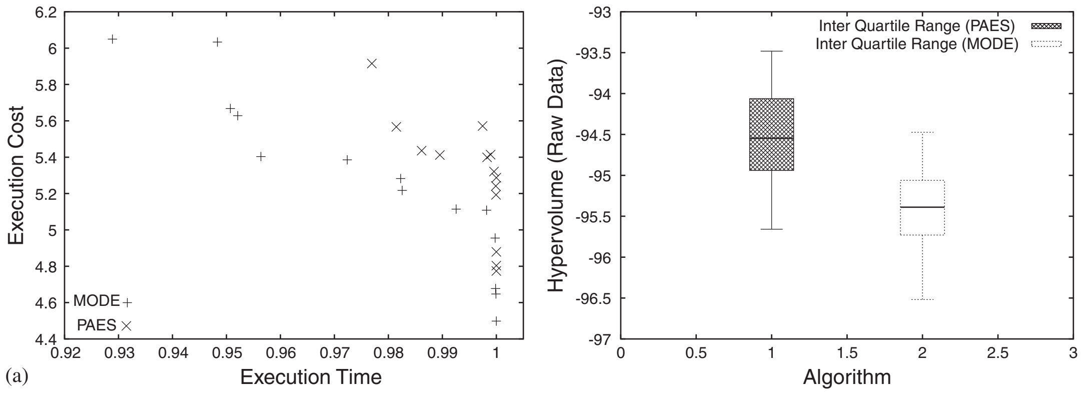
6. PERFORMANCE MEASURE

Generally, to find the comparative analysis between two MOEA, the Hypervolume (I_H) and Epsilon ($I_{\varepsilon+}$) indicators [35,36] are used. To find these values, we have run each algorithm for three different workflow structures [32,33]. The experiment for each scenario was repeated 30 times (with different random seeds). 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 across 30 runs. We then use the Hypervolume Difference indicator I_H^- [35] 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. The lower the value of I_H^-, the better the algorithm performs. Box plots in Figure 5(a), (b) and (c) (right side: I_H^- indicators for three different workflow examples) clearly prove that our approach outperforms PAES.

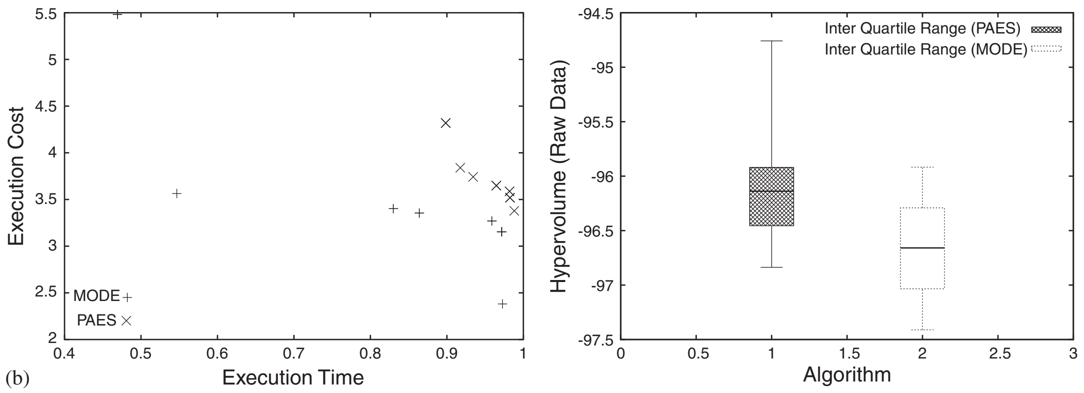
Although I_H is widely used in this context, however, it sometimes gives different values for two distributions of solutions even they are incomparable. To make more specific comparison, we

Table I. Selected workflow models.

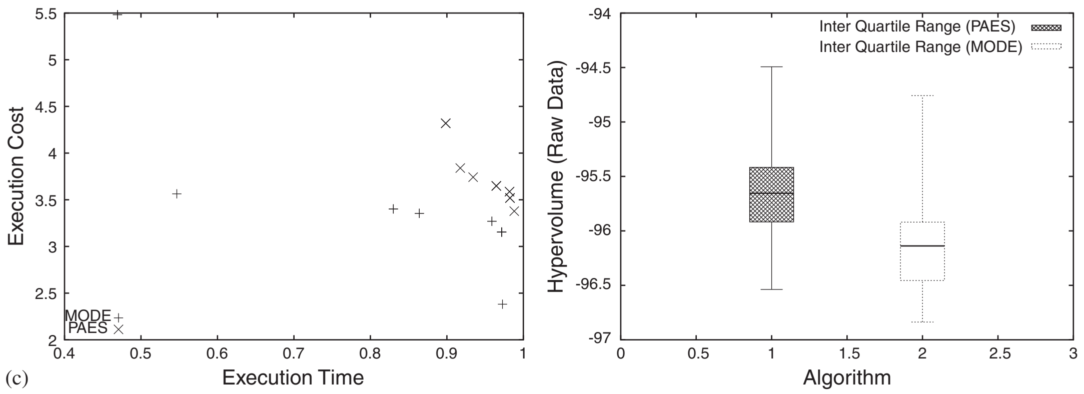
Workflow	# Nodes	Reference	Note
W1	15	[32]	Gauss–Jordan Algorithm
W2	14	[32]	LU decomposition
W3	16	[33]	Laplace Transform



(a)

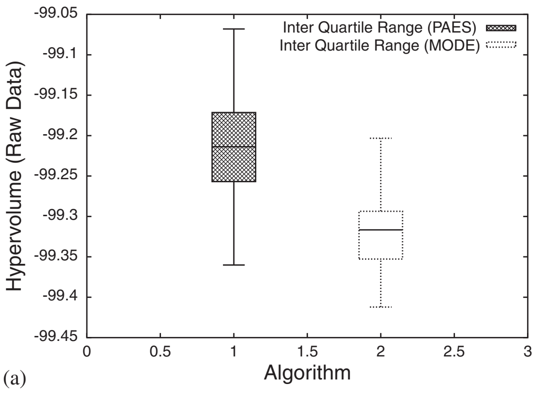


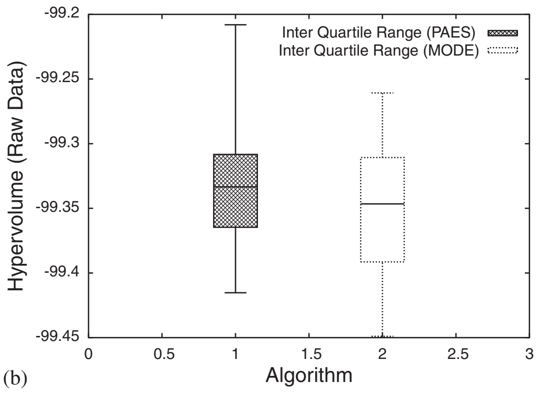
(b)



(c)

Figure 5. Non-dominated solutions and hypervolume measure (box plots) for three workflow benchmarks: (a) non-dominated solutions (left) and hypervolume measure (right) for workflow W1; (b) non-dominated solutions (left) and hypervolume measure (right) for workflow W2; and (c) non-dominated solutions (left) and hypervolume measure (right) for workflow W3.





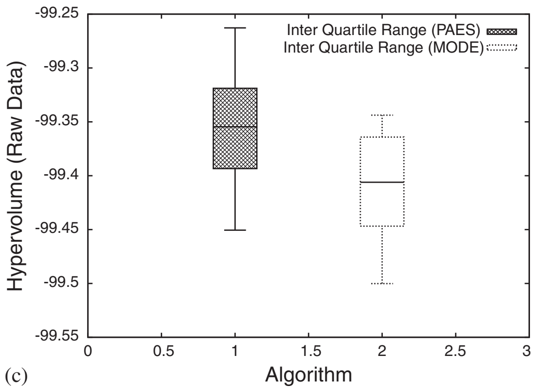


Figure 6. Epsilon measure (box plots) for three workflow benchmarks: (a) epsilon measure for workflow W1; (b) epsilon measure for workflow W2; and (c) epsilon measure for workflow W3.

need to use the Epsilon indicator ($I_{\varepsilon+}$). This indicator gives us the idea about how two separate distributions of Pareto-fronts differ from each other according to their degree of $\varepsilon+$–non-domination (for more details, please refer to [35]). We have also measured the $I_{\varepsilon+}$ values for the Pareto-front generated by two algorithms on the same benchmark workflow examples as I_H. For $I_{\varepsilon+}$, the lower value will indicate a better front. The Figure 6(a), (b) and (c) indicates that MODE is better than PEAS with respect to $I_{\varepsilon+}$ as well.

7. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a workflow execution planning approach, which optimizes 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. Our MODE approach differs from others in the sense that we are dealing with real scheduling sequences rather than job/machine sequence to real number transformation as implemented in [7,8]. We have compared our results with PAES and the I_H^- indicator shows that our approach performs better than PAES. Moreover, our model is also free from extra computational overhead due to crowding distance sorting. In the case of candidate creation, we have only considered the Ulam distance and there may be further opportunities to investigate the algorithms performance with different string similarity metrics.

Multiobjective optimization in Grid scheduling is not a matured field. It still requires a number of detailed benchmark problems to test every type of multiobjective scenario on Grid scheduling or real-life data to test an algorithm's performance. There are also a limited number of studies that have considered the flow-shop/job-shop scheduling problem using DE, however, multiobjective scheduling poses additional challenges. Many of the widely used existing workflow scheduling algorithm only attempt to minimize either execution time or execution cost. However, additional objectives must be considered when scheduling workflows on utility Grids.

REFERENCES

1. Foster I, Kesselman C. *The Grid*: *Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers: San Francisco, CA, 1998.
2. Deelman E. Mapping abstract complex workflows onto Grid environments. *Journal of Grid Computing* 2003; **1**:25–39.
3. Yu J, Buyya R, Ramamohanarao K. Workflow scheduling algorithms for Grid computing. *Metaheuristics for Scheduling in Distributed Computing Environments*, (*Studies in Computational Intelligence*), Xhafa F, Abraham A (eds.). Springer: Berlin, Heidelberg, 2008; 173–214.
4. He X, Sun XH, Laszewski GV. QoS guided min–min heuristic for Grid task scheduling. *Journal of Computer Science and Technology* 2003; **18**(4):442–451.
5. Wang L, Siegel HJ, Roychowdhury VP, Maciejewski AA. Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach. *Journal of Parallel Distributed Computing* 1997; **47**:9–22.
6. Wu AS, Yu H, Jin S, Lin K-C, Schiavone G. An incremental genetic algorithm approach to multiprocessor scheduling. *IEEE Transactions on Parallel and Distributed Systems* 2004; **15**(9):824–834.
7. Nearchou CA, Omirou SL. Differential evolution for sequencing and scheduling optimization. *Journal of Heuristics* 2005; **12**:395–411.
8. Onwubolu G, Davendra D. Scheduling flowshops using differential evolution algorithm. *European Journal of Operational Research* 2004; **171**:674–692.
9. Daoud M, Kharma N. GATS 1.0: A novel GA-based scheduling algorithm for task scheduling on heterogeneous processor nets. *GECCO '05*: *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*. ACM: New York, NY, U.S.A., 2005; 2209–2210.
10. Zhong Y-W, Yang J-G. A genetic algorithm for tasks scheduling in parallel multiprocessor systems. *International Conference on Machine Learning and Cybernetics*, X'ian, China, vol. 3, November 2003; 1785–1790.
11. Kim SC, Lee S. Push-pull: Guided search dag scheduling for heterogeneous clusters. *International Conference on Parallel Processing*, *ICPP 2005*, University of Oslo, Norway, June 2005; 603–610.
12. Woo S-H, Yang S-B, Kim S-D, Han T-D. Task scheduling in distributed computing systems with a genetic algorithm. *High Performance Computing on the Information Superhighway*, *HPC Asia '97*, Seoul, Korea, April–2 May 1997; 301–305.
13. Prodan R, Fahringer T. Dynamic scheduling of scientific workflow applications on the Grid: A case study. *SAC '05*: *Proceedings of the 2005 ACM Symposium on Applied Computing*, Santa Fe, New Mexico. ACM: New York, U.S.A., 2005; 687–694.
14. Wieczorek M, Prodan R, Fahringer T. Scheduling of scientific workflows in the ASKALON Grid environment. *SIGMOD Record* 2005; **34**(3):56–62.
15. Yu J, Buyya R. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Scientific Programming* 2006; **14**(3,4):217–230.
16. Wu M-Y, Shu W, Gu J. Efficient local search for DAG scheduling. *IEEE Transactions on Parallel and Distributed Systems* 2001; **12**(6):617–627.
17. Martino VD, Mililotti M. Scheduling in a Grid computing environment using genetic algorithms. *IPDPS*. IEEE Computer Society: Silver Spring, MD, 2002.

18. Martino VD, Mililotti M. Sub optimal scheduling in a Grid using genetic algorithms. *Parallel Computing* 2004; **30**(5–6):553–565.
19. Gao Y, Rong H, Huang JZ. Adaptive Grid job scheduling with genetic algorithms. *Future Generation Computer Systems* 2005; **21**(1):151–161.
20. Yao W, Xie X, You J. Link-contention-aware genetic scheduling using task duplication in Grid environments. *GCC* (*2*) (*Lecture Notes in Computer Science*, vol. 3033), Li M, Sun X-H, Deng Q, Ni J (eds.). Springer: Berlin, 2003; 822–829.
21. Kim S, Weissman JB. A genetic algorithm based approach for scheduling decomposable data Grid applications. *ICPP*. IEEE Computer Society: Silver Spring, MD, 2004; 406–413.
22. Dhodhi M, Hielscher F, Storer R, Bhasker J. Datapath synthesis using a problem-space genetic algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 1995; **14**(8):934–944.
23. Ye G, Rao R, Li M. A multiobjective resource scheduling approach based on genetic algorithms in Grid environment. *Fifth International Conference on Grid and Cooperative Workshops*, Hunan, China, 2006; 504–509.
24. Ishibuchi H, Murata T. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on System*, *Man*, *and Cybernetics—Part C*: *Applications and Reviews* 1998; **28**(3):392–403.
25. Braun TD, Siegel HJ, Beck N, Bölöni M, Maheswaran M, Reuther AI, Robertson JP, Theys MD, Yao B, Hensgen DA, Freund RF. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing* 2001; **61**(6):810–837.
26. Abraham A, Buyya R. Nature's heuristics for scheduling jobs on computational Grids. *Proceedings of the 8th International Conference on Advanced Computing and Communications* (*ADCOM 2000*), Cochin, India. Tata McGraw-Hill Publishing Co. Ltd.: New Delhi, India, 2000; 45–52.
27. Pan Q-K, Tasgetiren MF, Liang YC. A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Genetic and Evolutionary Computation Conference*. ACM Press: New York, 2007; 126–133.
28. Sarker R, Abbass HA. Differential evolution for solving multi-objective optimization problems. *Asia–Pacific Journal of Operations Research* 2004; **21**(2):225–240.
29. Storn R, Price K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 1997; **11**(4):241–354.
30. Yu J, Kirley M, Buyya R. Multi-objective planning for workflow execution on Grids. *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*, Singapore. IEEE Press: Piscataway, NJ, September 2007; 10–17.
31. Bhat DN. An evolutionary measure for image matching. *Fourteenth International Conference on Pattern Recognition*. IEEE Press: New York, 1998; 850–852.
32. Tsuchiya T, Osada T, Kikuno T. Genetic-based multiprocessor scheduling using task duplication. *Microprocessors and Microsystems* 1998; **22**:197–207.
33. Wu M, Gajski D. Hypertool: A programming aid for message-passing systems. *IEEE Transactions on Parallel and Distributed Systems* 1990; **1**(3):330–343.
34. Knowles J, Corne D. The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation. *IEEE Congress on Evolutionary Computation*. IEEE Press: New York, 1999; 98–105.
35. Knowles J, Thiele L, Zitzler E. A tutorial on the performance assessment of stochastic multiobjective optimizers. *Technical Report 214*, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, February 2006, Revised Version.
36. Zitzler E, Thiele L, Laumanns M, Fonseca CM, da Fonseca V. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation* 2003; **7**(2):117–132.