1

w-TG: A Combined Algorithm to Optimize the Runtime of the Grid-Based Workflow Within an SLA Context

Dang Minh Quan[1], Joern Altmann[2] and Laurence T. Yang[3]
[1]*Center for REsearch And Telecommunication Experimentation for NETworked Communities*
[2]*Technology Management, Economics, and Policy Program, Department of Industrial Engineering, College of Engineering, Seoul National University*
[3]*Department of Computer Science, St. Francis Xavier University*
[1]*Italy*
[2]*South Korea*
[3]*Canada*

1. Introduction

In the Grid Computing environment, many users need the results of their calculations within a specific period of time. Examples of those users are weather forecasters running weather forecasting workflows, and automobile producers running dynamic fluid simulation workflow Lovas et al. (2004). Those users are willing to pay for getting their work completed on time. However, this requirement must be agreed on by both, the users and the Grid provider, before the application is executed. This agreement is contained in the Service Level Agreement (SLA) Sahai et al. (2003). In general, SLAs are defined as an explicit statement of expectations and obligations in a business relationship between service providers and customers. SLAs specify the a-priori negotiated resource requirements, the quality of service (QoS), and costs. The application of such an SLA represents a legally binding contract. This is a mandatory prerequisite for the Next Generation Grids.

However, letting Grid-based workflows' owners work directly with resource providers has two main disadvantages:

- The user has to have a sophisticated resource discovery and mapping tools in order to find the appropriate resource providers.
- The user has to manage the workflow, ranging from monitoring the running process to handling error events.

To free users from this kind of work, it is necessary to introduce a broker to handle the workflow execution for the user. We proposed a business model Quan & J. Altmann (2007) for the system as depicted in Figure 1, in which, the SLA workflow broker represents the user as specified in the SLA with the user. This controls the workflow execution. This includes

mapping of sub-jobs to resources, signing SLAs with the services providers, monitoring, and error recovery. When the workflow execution has finished, it settles the accounts, pays the service providers and charges the end-user. The profit of the broker is the difference. The value-added that the broker provides is the handling of all the tasks for the end-user.

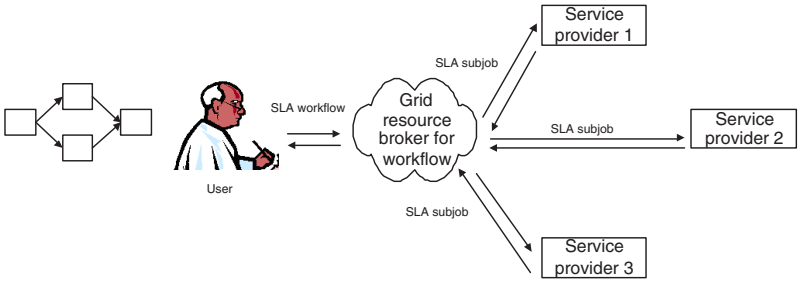


Fig. 1. Stakeholders and their business relationship

We presented a prototype system supporting SLAs for the Grid-based workflow in Quan et al. (2005; 2006); Quan (2007); Quan & Altmann (2007). Figure 2 depicts a sample scenario of running a workflow in the Grid environment.

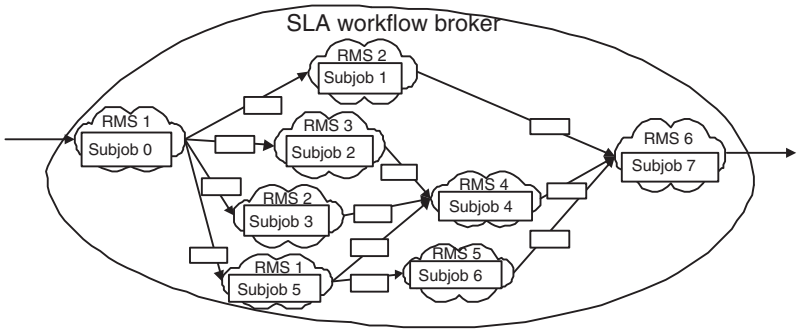


Fig. 2. A sample running Grid-based workflow scenario

In the system handling the SLA-based workflow, the mapping module receives an important position. Our ideas about Grid-based workflow mapping within the SLA context have 3 main scenarios.

- Mapping heavy communication Grid-based workflow within the SLA context, satisfying the deadline and optimizing the cost Quan et al. (2006).
- Mapping light communication Grid-based workflow within the SLA context, satisfying the deadline and optimizing the cost Quan & Altmann (2007).
- Mapping Grid-based workflow within the SLA context with execution time optimization.

The requirement of optimizing the execution time emerges in several situations.

- In the case of catastrophic failure, when one or several resource providers are detached from the grid system at a time, the ability to finish the workflow execution on time as

stated in the original SLA is very low and the ability to be fined because of not fulfilling SLA is nearly 100%. Within the SLA context, which relates to business, the fine is usually very high and increases with the lateness of the workflow's finished time. Thus, those sub-jobs, which form an workflow, must be mapped to the healthy RMSs in a way, which minimizes the workflow finishing time Quan (2007).

- When the Grid is busy, there are few free resources. In this circumstance, finding a feasible solution meeting the user's deadline is a difficult task. This constraint equals to find an optimizing workflow execution time mapping solution. Even when the mapping result does not meet the preferred deadline, the broker can still use it for further negotiation with the user.

The previous work proposed an algorithm, namely the w-Tabu Quan (2007), to handle this problem. In the w-Tabu algorithm, a set of referent solutions, which distribute widely over the search space, is created. From each solution in the set, we use the Tabu search to find the local minimal solution. The Tabu search extends the local search method by using memory structures. When a potential solution has been determined, it is marked as "taboo" so that the algorithm does not visit that solution frequently. However, this mechanism only searches the area around the referent solution. Thus, many areas containing good solutions may not be examined by the w-Tabu algorithm and thus, the quality of the solution is still not as high as it should be.

In this book chapter, we propose a new algorithm to further improve the quality of the mapping solution. The main contribution of the book chapter includes:

- An algorithm based Genetic algorithm called the w-GA algorithm. According to the character of the workflow, we change the working mechanism of the crossover and mutation operations. Thus, the algorithm could find a better solution than the standard GA algorithm with the same runtime.
- An analysis the strong and weak points of w-GA algorithm compared to the w-Tabu algorithm. We do an extensive experiment in order to see the quality of w-GA algorithm in performance and runtime.
- An combined algorithm, namely w-TG. We propose a new algorithm by combining the w-GA algorithm and the w-Tabu algorithm. The experiment shows that the new algorithm finds out solutions about 9% greater than the w-Tabu algorithm.

In the early state of the business Grid like now, there are not so many users or providers and the probability of numerous requests coming at a time is very low. Moreover, even when the business Grid becomes crowd, there are many periods that only one SLA workflow request coming at a time. Thus, in this book chapter, we assume the broker handles one workflow running request at a time. The extension of mapping many workflows at a time will be the future work.

The chapter is organized as follows. Sections 2 and 3 describe the problem and the related works respectively. Section 4 presents the w-GA algorithm. Section 5 describes the performance experiment, while section 6 introduces the combined algorithm w-TG and its performance. Section 7 concludes the book chapter with a short summary.

Sjs	cpu	speed (Mhz)	stor (GB)	exp	rt (slot)	S-sj	D-sj	data (GB)
0	128	1000	30	2	16	0	1	1
1	64	1000	20	1	13	0	2	4
2	128	1000	30	2	14	0	3	6
3	128	1000	30	2	5	0	5	10
4	128	1000	30	2	8	1	7	3
5	32	1000	10	0	13	2	4	2
6	64	1000	20	1	16	3	4	8
7	128	1000	30	2	7	4	7	4
						5	4	3
						5	6	6
Workflow starting slot: 10						6	7	6

Table 1. Sample workflow specification

2. Problem statement

2.1 Grid-based workflow model

Like many popular systems handling Grid-based workflows Deelman et al. (2004); Lovas et al. (2004); Spooner et al. (2003), our system is of the Directed Acyclic Graph (DAG) form. The user specifies the required resources needed to run each sub-job, the data transfer between sub-jobs, the estimated runtime of each sub-job, and the expected runtime of the whole workflow. In this book chapter, we assume that time is split into slots. Each slot equals a specific period of real time, from 3 to 5 minutes. We use the time slot concept in order to limit the number of possible start-times and end-times of sub-jobs. Moreover, a delay of 3 minutes is insignificant for the customer. Table 1 presents the main parameters including sub-job specifications and data transfer specifications of the sample workflow in Figure 2. The sub-job specification includes the number of CPU (cpu), the CPU speed (speed), the amount of storage (stor), the number of experts (exp), the required runtime (rt). The data transfer specification includes the source sub-job (S-sj), the destination sub-job (D-sj), and the number of data (data). It is noted that the CPU speed of each sub-job can be different. However, we set it to the same value for the presentation purposes only.

2.2 Grid service model

The computational Grid includes many High Performance Computing Centers (HPCCs). The resources of each HPCC are managed by a software called local Resource Management System (RMS)[1]. Each RMS has its own unique resource configuration, the number of CPUs, the amount of memory, the storage capacity, the software, the number of experts, and the service price. To ensure that the sub-job can be executed within a dedicated time period, the RMS must support an advance resource reservation such as CCS Hovestadt (2003). Figure 3 depicts an example of an CPU reservation profile of such an RMS. In our model, we reserve three main types of resources: CPU, storage, and expert. The addition of further resources is straightforward.

[1] In this book chapter, RMS is used to represent the cluster/super computer as well as the Grid service provided by the HPCC.

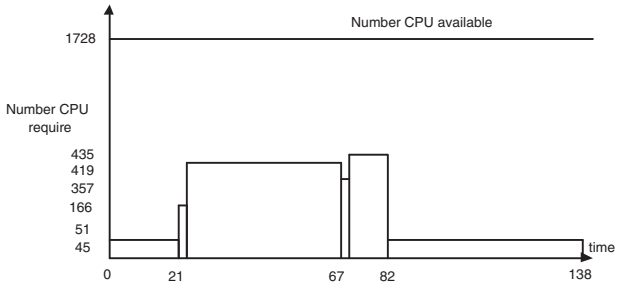


Fig. 3. A sample CPU reservation profile of a local RMS

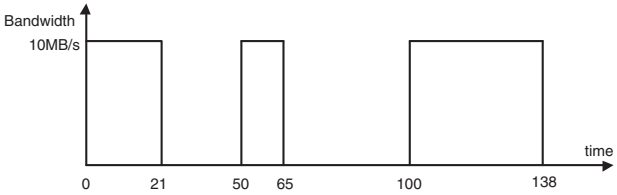


Fig. 4. A sample bandwidth reservation profile of a link between two local RMSs

If two output-input-dependent sub-jobs are executed on the same RMS, it is assumed that the time required for the data transfer equals zero. This can be assumed since all compute nodes in a cluster usually use a shared storage system such as NFS or DFS. In all other cases, it is assumed that a specific amount of data will be transferred within a specific period of time, requiring the reservation of bandwidth.

The link capacity between two local RMSs is determined as the average available capacity between those two sites in the network. The available capacity is assumed to be different for each different RMS couple. Whenever a data transfer task is required on a link, the possible time period on the link is determined. During that specific time period, the task can use the entire capacity, and all other tasks have to wait. Using this principle, the bandwidth reservation profile of a link will look similar to the one depicted in Figure 4. A more realistic model for bandwidth estimation (than the average capacity) can be found in Wolski (2003). Note, the kind of bandwidth estimation model does not have any impact on the working of the overall mechanism.

Table 2 presents the main resource configuration including the RMS specification and the bandwidth specification of the 6 RMSs in Figure 2. The RMS specification includes the number of CPU (cpu), the CPU speed in Mhz (speed), the amount of storage in GB (stor), the number of expert (exp). The bandwidth specification includes the source RMS (s), the destination RMS (d), and the bandwidth in GB/slot (bw). For presentation purpose, we assume that all reservation profiles are empty. It is noted that the CPU speed of each RMS can be different. We set it to the same value for the presentation purposes only.

2.3 Problem specification

The formal specification of the described problem includes following elements:

ID	cpu	speed	stor	exp	s	d	bw	s	d	bw
1	256	1000	3000	4	1	2	1	2	1	1
2	128	1000	2000	3	1	3	3	3	1	3
3	256	1000	3000	4	1	4	2	4	1	2
4	256	1000	3000	4	1	5	3	5	1	3
5	256	1000	3000	4	1	6	2	6	1	2
6	64	1000	1000	2	2	3	1	3	2	1
					2	4	1	4	2	1
					2	5	3	5	2	3
					2	6	2	6	2	2
					3	4	1	4	3	1
					3	5	3	5	3	3
					3	6	1	6	3	1
					4	5	2	5	4	2
					4	6	3	6	4	3
					5	6	1	6	5	1

Table 2. Sample RMS configurations

- Let R be the set of Grid RMSs. This set includes a finite number of RMSs, which provide static information about controlled resources and the current reservations/assignments.
- Let S be the set of sub-jobs in a given workflow including all sub-jobs with the resource and runtime requirements.
- Let E be the set of edges in the workflow, which express the dependency between the sub-jobs and the necessity for data transfers between the sub-jobs.
- Let K_i be the set of resource candidates of sub-job s_i. This set includes all RMSs, which can run sub-job s_i, $K_i \subset R$.

Based on the given input, the required solution includes two sets defined in Formula 1 and 2.

$$M = \{(s_i, r_j, start, stop) | s_i \in S, r_j \in K_i\} \tag{1}$$

$$N = \{(e_{ik}, start, stop) | e_{ik} \in E\} \tag{2}$$

If the solution does not have a start, stop slot for each s_i, it becomes a configuration as defined in Formula 3.

$$a = \{(s_i, r_j) | s_i \in S, r_j \in K_i\} \tag{3}$$

A feasible solution must satisfy following conditions:

- **Criterion 1:** All $K_i \neq \emptyset$. There is at least one RMS in the candidate set of each sub-job.
- **Criterion 2:** The dependencies of the sub-jobs are resolved and the execution order remains unchanged.
- **Criterion 3:** The capacity of an RMS must equal or greater than the requirement at any time slot. Each RMS provides a profile of currently available resources and can run many sub-jobs of a single flow both sequentially and in parallel. Those sub-jobs, which run on the same RMS, form a profile of resource requirement. With each RMS r_j running sub-jobs of the Grid workflow, with each time slot in the profile of available resources and profile of

resource requirements, the number of available resources must be larger than the resource requirement.

- **Criterion 4:** The data transmission task e_{ki} from sub-job s_k to sub-job s_i must take place in dedicated time slots on the link between the RMS running sub-job s_k to the RMS running sub-job s_i. $e_{ki} \in E$.

The goal is to minimize the makespan of the workflow. The makespan is defined as the period from the desired starting time until the finished time of the last sub-job in the workflow. In addition to the aspect that the workflow in our model includes both parallel and sequential sub-jobs, the SLA context imposes the following distinguishing characteristics.

- An RMS can run several parallel or sequential sub-jobs at a time.
- The resources in each RMS are reserved.
- The bandwidth of the links connecting RMSs is reserved.

To check for the feasibility of a configuration, the mapping algorithm must go through the resource reservation profiles and bandwidth reservation profile. This step needs a significant amount of time. Suppose, for example, that the Grid system has m RMS, which can satisfy the requirement of n sub-jobs in a workflow. As an RMS can run several sub-jobs at a time, finding out the optimal solution needs (m^n) loops for checking the feasibility. It can be easily shown that the optimizing of the execution time of the workflow on the Grid as described above is an NP hard problem Black et al. (1999). Previous experiment results have shown that with the number of sub-jobs equaling 6 and number of RMSs equaling 20, the runtime to find out the optimal solution is exponential Quan et al. (2007).

3. Related works

The mapping algorithm for Grid workflow has received a lot of attentions from the scientific community. In the literature, there are many methods to mapping a Grid workflow to Grid resource within different contexts. Among those, the old but well-known algorithm Condor-DAGMan from the work of Condor (2004) is still used in some present Grid systems. This algorithm makes local decisions about which job to send to which resource and considers only jobs, which are ready to run at any given instance. Also, using a dynamic scheduling approach, Duan et al. (2006) and Ayyub et al. (2007) apply many techniques to frequently rearrange the workflow and reschedule it in order to reduce the runtime of the workflow. Those methods are not suitable for the context of resource reservation because whenever a reservation is canceled, a fee is charged. Thus, frequent rescheduling may lead to a higher running workflow cost.

Deelman et al. (2004) presented an algorithm which maps Grid workflows onto Grid resources based on existing planning technology. This work focuses on coding the problem to be compatible with the input format of specific planning systems and thus transferring the mapping problem to a planning problem. Although this is a flexible way of gaining different destinations, which includes some SLA criteria, significant disadvantages regarding the time-intensive computation, long response times and the missing consideration of Grid-specific constraints appeared.

In Mello et al. (2007), Mello et. al. describe a load balancing algorithm addressed to Grid computing environment called RouteGA. The algorithm uses GA techniques to provide an

equal load distribution based on the computing resources capacity. Our work is different from the work of Mello et. al. in two main aspects.

- While we deal with workflow, the work in Mello et al. (2007) considers a group of single jobs but with no dependency among them.
- In our work, The resources are reserved, whereas Mello et al. (2007) does not consider the resource reservation context.

Related to the mapping task graph to resources, there is also the multiprocessor scheduling precedence-constrained task graph problem Gary et al. (1979); Kohler et al. (1974). As this is a well-known problem, the literature has recorded a lot of methods for this issue, which can be classified into several groups Kwok et al. (1999). The classic approach is based on the so-called list scheduling technique Adam et al. (1974); Coffman et al. (1976). More recent approaches are the UNC (Unbounded Number of Clusters) Scheduling Gerasoulis et al. (1992); Sarkar (1989), the BNP (Bound Number of Processors) Scheduling Adam et al. (1974); Kruatrachue et al. (1987); Sih et al. (1993), the TDB (Task Duplication Based) Scheduling Colin et al. (1991); Kruatrachue et al. (1988), the APN (Arbitrary Processor Network) Scheduling Rewini et al. (1990), and the genetic Hou et al. (1994); Shahid et al. (1994). Our problem differs from the multiprocessor scheduling precedence-constrained task graph problem in many factors. In the multiprocessor scheduling problem, all processors are similar, but in our problem, RMSs are heterogeneous. Each task in our problem can be a parallel program, while each task in the other problem is a strictly sequential program. Each node in the other problem can process one task at a time while each RMS in our problem can process several sub-jobs at a time. For these reasons, we cannot apply the proposed techniques to our problem because of the characteristic differences.

In recent works Berman et al. (2005); Blythe et al. (2005); Casanova et al. (2000); Ma et al. (2005), authors have described algorithms which concentrate on scheduling the workflow with parameter sweep tasks on Grid resources. The common destination of those algorithms is optimizing the makespan, defined as the time from when execution starts until the last job in the workflow is completed. Subtasks in this kind of workflow can be group in layers and there is no dependency among subtasks in the same layer. All proposed algorithms assume each task as a sequential program and each resource as a compute node. By using several heuristics, all those algorithms perform the mapping very quickly. Our workflow with the DAG form can also be transformed to the workflow with parameter sweep tasks type, and thus we have applied all those algorithms to our problem.

Min-min algorithm

Min-min uses the Minimum MCT (Minimum Completion Time) as a measurement, meaning that the task that can be completed the earliest is given priority. The motivation behind Min-min is that assigning tasks to hosts that will execute them the fastest will lead to an overall reduced finished time Berman et al. (2005); Casanova et al. (2000). To adapt the min-min algorithm to our problem, we analyze the workflow into a set of sub-jobs in sequential layers. Sub-jobs in the same layer do not depend on each other. With each sub-job in the sequential layer, we find the RMS which can finish sub-job the earliest. The sub-job in the layer which has the earliest finish time, then, will be assigned to the determined RMS. A more detailed description about the algorithm can be seen in Quan (2007).

Max-min algorithm

Max-min's metric is the Maximum MCT. The expectation is to overlap long-running tasks with short-running ones Berman et al. (2005); Casanova et al. (2000). To adapt the max-min algorithm to our problem, we analyze the workflow into a set of sub-jobs in sequential layers. Sub-jobs in the same layer do not depend on each other. With each sub-job in the sequential layer, we find the RMS which can finish sub-job the earliest. The sub-job in the layer which has the latest finish time, will be assigned to the determined RMS. A more detailed description about the algorithm can be seen in Quan (2007).

Suffer algorithm

The rationale behind sufferage is that a host should be assigned to the task that would "suffer" the most if not assigned to that host. For each task, its sufferage value is defined as the difference between its best MCT and its second-best MCT. Tasks with a higher sufferage value take precedence Berman et al. (2005); Casanova et al. (2000). To adapt a suffer algorithm to our problem, we analyze the workflow into a set of sub-jobs in sequential layers. Sub-jobs in the same layer do not depend on each other. With each sub-job in the sequential layer, we find the earliest and the second-earliest finish time of the sub-job. The sub-job in the layer which has the highest difference between the earliest and the second-earliest finish time will be assigned to the determined RMS. A more detailed description about the algorithm can be seen in Quan (2007).

GRASP algorithm

In this approach a number of iterations are made to find the best possible mapping of jobs to resources for a given workflow Blythe et al. (2005). In each iteration, an initial allocation is constructed in a greedy phase. The initial allocation algorithm computes the tasks whose parents have already been scheduled on each pass, and consider every possible resource for each such task. A more detailed description about the algorithm can be seen in Quan (2007).

w-DCP algorithm

The DCP algorithm is based on the principle of continuously shortening the longest path (also called critical path (CP)) in the task graph by scheduling tasks in the current CP to an earlier start time. This principal was applied for scheduling workflows with parameter sweep tasks on global Grids by Tianchi Ma et al in Ma et al. (2005). We proposed a version of DCP algorithm to our problem in Quan (2007).

The experiment results show that the quality of solutions found by those algorithm is not sufficient Quan (2007). To overcome the poor performance of methods in the literature, in the previous work Quan (2007), we proposed the w-Tabu algorithm. An overview of w-Tabu algorithm is presented in Algorithm 1.

The assigning sequence is based on the latest start_time of the sub-job. Sub-jobs having smaller latest start time will be assigned earlier. Each solution in the reference solutions set can be thought of as the starting point for the local search so it should be spread as widely as possible in the searching space. To satisfy the space spread requirement, the number of similar map $sub - job : RMS$ between two solutions, must be as small as possible. The improvement procedure based on the Tabu search has some specific techniques to reduce the computation time. More information about w-Tabu algorithm can be seen in Quan (2007).

Algorithm 1 w-Tabu algorithm

1: Determine assignning sequence for all sub-jobs of the workflow
2: Generate reference solution set
3: **for all** solution in reference set **do**
4: Improve the solution as far as possible with the modified Tabu search
5: **end for**
6: Pick the solution with best result

4. w-GA algorithm

4.1 Standard GA

The standard application of GA algorithm to find the minimal makespan of a workflow within an SLA context is presented in Algorithm 2. We call it the n-GA algorithm.

Algorithm 2 n-GA algorithm

1: Determine assigning sequence for all sub-jobs of the workflow
2: Generate reference configuration set
3: **while** $num\_mv < max$ **do**
4: Evaluate the *makespan* of each configuration
5: a"= best configuration
6: Add a" to the new population
7: **while** the new population is not enough **do**
8: Select parent couple configurations according to their *makespan*
9: Crossover the parent with a probability to form new configurations
10: Mutate the new configuration with a probability
11: Put the new configuration to the new population
12: **end while**
13: $num\_mv \leftarrow num\_mv + 1$
14: **end while**
15: return a"

Determining the assigning sequence

The sequence of determining runtime for sub-jobs of the workflow in an RMS can also affect the final makespan, especially in the case of many sub-jobs in the same RMS. Similar to w-Tabu algorithm, the assigning sequence is based on the latest *start_time* of the sub-job. Sub-jobs having the smaller latest start time will be assigned earlier. The complete procedure can be seen in Quan (2007). Here we outline some main steps. We determine the earliest and the latest start time for each of the sub-jobs of the workflow under ideal conditions. The time period to do data transferring among sub-jobs is computed by dividing the amount of data over a fixed bandwidth. The latest start/stop time for each sub-job and each data transfer depends only on the workflow topology and the runtime and not on the resources context. Those parameters can be determined by using conventional graph algorithms.

Generating the initial population

In the n-GA algorithm, the citizen is encoded as described in Figure 5. We use this convention encoding as it naturally presents a configuration and thus, it is very convenient to evaluate the timetable of the solution.

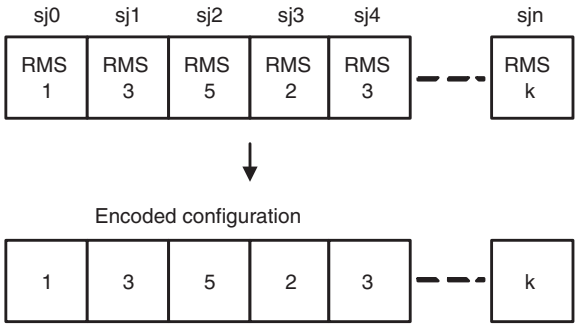


Fig. 5. The encoded configuration

Each sub-job has different resource requirements and there are a lot of RMSs with different resource configurations. The initial action is finding among those heterogeneous RMSs the suitable RMSs, which can meet the requirement of the sub-job. The matching between the sub-job's resource requirement and the RMS's resource configuration is done by several logic checking conditions in the WHERE clause of the SQL SELECT command. This work will satisfy Criterion 1. The set of candidate lists is the configuration space of the mapping problem.

The crossover operation of the GA will reduce the distance between two configurations. Thus, to be able to search over a wide search area, the initial population should be distributed widely. To satisfy the space spreading requirement, the number of the same map sub-job:RMS between two configurations must be as small as possible. We apply the same algorithm for creating the initial set of the configuration in Quan (2007). The number of the member in the initial population set depends on the number of available RMSs and the number of sub-jobs.

For example, from Table 1 and 2, the configuration space of the sample problem is presented in Figure 6a. The initial population will be presented in Figure 6b.

Determining the makespan

The fitness value is based on the makespan of the workflow. In order to determine the makespan of a citizen, we have to calculate the timetable of the whole workflow. The algorithm for computing the timetable is presented in Algorithm 3. The start and stop time of the sub-job is determined by searching the resource reservation profile. The start and stop time of data transfer is determined by searching the bandwidth reservation profile. This procedure will satisfy Criteria 2 and 3 and 4.

After determining the timetable, we have a solution. With our sample workflow, the solution of the configuration 1 in Figure 6b including the timetable for sub-jobs and the time table for data transfer is presented in Table 3. The timetable for sub-jobs includes the RMS and the start, stop time of executing the sub-job. The timetable for data transfer includes the source and destination sub-jobs (S-D sj), source and destination RMS (S-D rms), and the start and stop time of performing the data transfer. The makespan of this sample solution is 64.

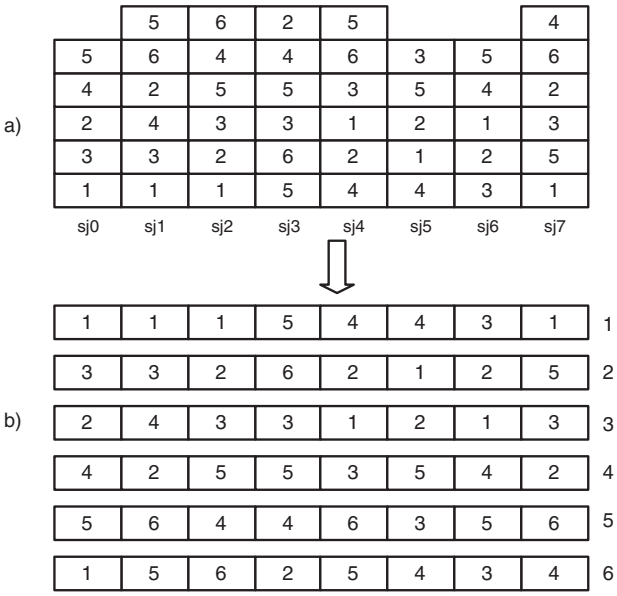


Fig. 6. Sample of forming the initial population

Algorithm 3 Determining timetable algorithm

1: **for** Each sub-job k following the assign sequence **do**
2: Determine set of assigned sub-jobs Q, which having output data transfer to the sub-job k
3: **for** Each sub-job i in Q **do**
4: min_st_tran=end_time of sub-job i +1
5: Search in reservation profile of link between RMS running sub-job k and RMS running sub-job i to determine start and end time of data transfer task with the start time > min_st_tran
6: **end for**
7: min_st_sj=max end time of all above data transfer +1
8: Search in reservation profile of RMS running sub-job k to determine its start and end time with the start time > min_st_sj
9: **end for**

Crossover and mutation

Parents are selected according to the roulette wheel method. The fitness of each configuration = $1/makespan$. Firstly, the sum L of all configuration fitness is calculated. Then, a random number l from the interval $(0, L)$ is generated. Finally, we go through the population to sum the fitness p. When p is greater than l, we stop and return to the configuration where we were.

The crossover point is chosen randomly. For the purpose of demonstration, we use the sample workflow in Figure 2. Assume that we have a parents and a crossover point as presented in Figure 7a. The child is formed by copying from two parts of the parents. The result is presented in Figure 7b. The mutation point is chosen randomly. At the mutation point, r_j of s_i

Sjs	RMS	start-stop	S-D sj	S-D rms	start-stop
0	1	10-25	0-1	1-1	0-0
1	1	26-29	0-2	1-1	0-0
2	1	30-42	0-3	1-5	26-27
3	5	28-32	0-5	1-4	26-30
4	4	44-51	1-7	1-1	0-0
5	4	31-43	2-4	1-4	43-43
6	3	50-65	3-4	5-4	33-36
7	1	68-74	4-7	4-1	52-53
			5-4	4-4	0-0
			5-6	4-3	44-49
			6-7	3-1	66-67

Table 3. Sample solution timetable

is replaced by another RMS in the candidate RMS set. It is noted that the probability of having a mutation with a child is low, ranging approximately from 0.5% to 1%. The final result is presented in Figure 7c.

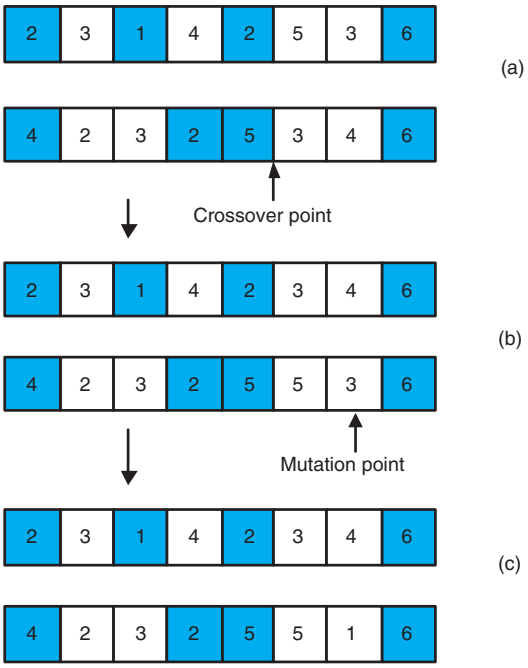


Fig. 7. Standard GA operations

4.2 Elimination of the standard GA

We did several experiments with n-GA and the initial result was not satisfactory. The algorithm has long runtime and presents low quality solutions. We believe that the reason

for this is located in we do the crossover and mutation operations. In particular, we do not carefully consider the critical path of the workflow. The runtime of the workflow depends mainly on the execution time of the critical path. With a defined solution and timetable, the critical path of a workflow is defined with the algorithm as described in Algorithm 4.

Algorithm 4 Determining critical path algorithm

1: Let C is the set of sub-jobs in the critical path
2: Put last sub-job into C
3: *next_subjob*=last sub-job
4: **repeat**
5: *prev_subjob* is determined as the sub-job having latest finished data output transfer to *next_subjob*
6: Put *prev_subjob* into C
7: $next\_sj = prev\_subjob$
8: **until** *prev_sj*= first sub-job

We start with the last sub-job determined. The next sub-job of the critical path will have the latest finish data transfer to the previously determined sub-job. The process continues until the next sub-job becomes the first sub-job.

The purpose of the crossover operation in the n-GA algorithm is creating new solutions in the hope that they are superior to the old one. In the crossover phase of the GA algorithm, when the sub-jobs of the critical path are not moved to other RMSs, the old critical path will have very low probability of being shortened. Thus, the overall makespan of the workflow has a low probability of improvement.

The primary purpose of the mutation operation is to maintain genetic diversity from one generation of a population of chromosomes to the next. In particular, it allows the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other. When the sub-jobs of the critical path are not moved to other RMSs, the old critical path will have very low probability of being changed. Thus, the mutation operation does not have as good effect as it should have.

With the standard GA algorithm, it is always possible that the above situation happens and thus creates a long convergent process. We can see an example scenario in Figure 7. Assume that we select a parent as presented in Figure 7a. Using the procedure in Algorithm 4, we know the critical path of each solution which is marked by colour boxes. After the crossover and mutation operation as described in Figure 7b, 7c, the old critical path remains the same.

To overcome this elimination, we propose an algorithm called the w-GA algorithm.

4.3 w-GA algorithm

The framework of the w-GA algorithm is similar to the n-GA algorithm. We focus on the crossover and mutation operations. Assume that we selected a parent such as in Figure 7(a), the following steps will be taken.

Step 1: Determine the critical path of each solution. The procedure to determine this is described in Algorithm 4. In each solution of our example, the sub-jobs joined with the critical

path are marked with color. The sub-jobs joined with the critical path in solution 1 include 0, 2, 4, 7. The sub-jobs joined the critical path in solution 2 include 0, 3, 4, 7.

Step 2: Form the critical set. The critical set includes sub-jobs have appeared in both critical paths. With our example, the critical set includes sub-jobs 0, 2, 3, 4, 7.

Step 3: Create derived configurations. The derived configuration is extracted from the old one by getting only sub-jobs which have appeared in the critical set. After this step, the two new configurations of the example are presented in Figure 8.

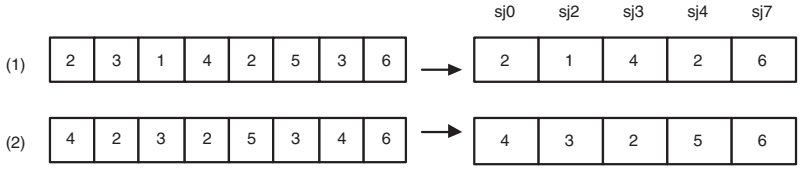


Fig. 8. The new derived configurations

Step 4: Exchange assignment if there is improvement signal. A couple $(s_i : r_j | s_i \in S, r_j \in K_i)$ is called an assignment. Assume that $(s1_i : r_j)$ is an assignment of the derived configuration 1, and $(s2_i : r_k)$ is an assignment of the derived configuration 2. If we change $(s1_i : r_j)$ to $(s1_i : r_k)$ and the finished time of the data transfer from the sub-job $s1_i$ to the next sub-job in the critical path is decreased, we say that the improvement signal appears. Without the improvement signal, the critical path cannot be shortened and the *makespan* cannot be improved. The algorithm for doing the exchange assignment is presented in Algorithm 5.

Algorithm 5 Exchange assignment algorithm

```
1: imp_signal ← 0
2: for each sub-job s_i in the critical set do
3:     (s1_i : r_j) change to (s1_i : r_k)
4:     if has improving signal then
5:         imp_signal ← 1
6:     else
7:         (s1_i : r_k) change to (s1_i : r_j)
8:     end if
9:     (s2_i : r_k) change to (s2_i : r_j)
10:    if has improving signal then
11:        imp_signal ← 1
12:    else
13:        (s1_i : r_j) change to (s1_i : r_k)
14:    end if
15: end for
```

With each sub-job, we exchange the RMS between two configurations. If the exchange indicates an improvement signal, we keep the change. Otherwise, we return to the old assignment.

If there are some changes in either of the two configurations, we move to step 5. If there is no change, we move to step 4. In our example, the possible changes could be presented in Figure 9.

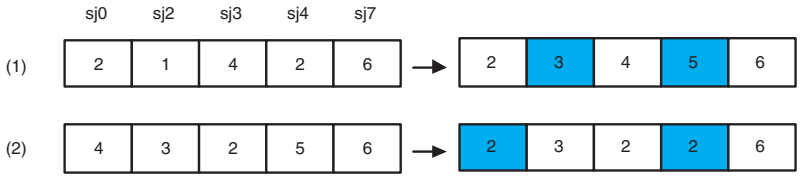


Fig. 9. Derived configurations after exchanging

Step 5: Do crossover. When there is no change with step 4 we do a normal crossover with the two derived configurations. This procedure is presented in Figure 10.

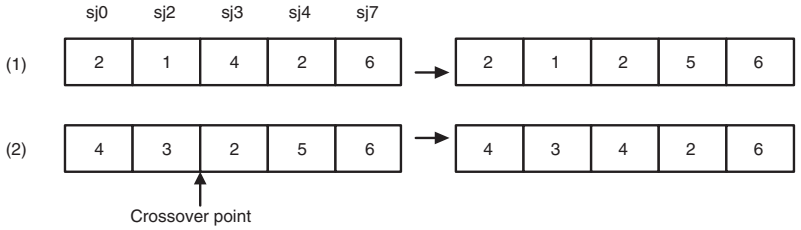


Fig. 10. Normal crossover operations

Step 6: Do mutation. The mutation is done on the derived configuration with no successful change. With each selected configuration, the mutation point is chosen randomly. At the mutation point, r_j of s_i is replaced by another RMS in the candidate RMS set. Like the normal GA algorithm, the probability to do mutation with a configuration is small. We choose a random selection because the main purpose of mutation is to maintain genetic diversity from one generation of a population of chromosomes to the next. If we also use mutation to improve the quality of the configuration, the operation mutation needs a lot of time. Our initial experiment shows that the algorithm cannot find a good solution within the allowable period.

Step 7: Reform the configuration. We return the derived configurations to the original configurations to have the new configurations. With our example, assume that step 4 is successful so we have two new derived configurations as in Figure 9. The new configurations are presented in Figure 11.

5. w-GA performance and discussion

The goal of the experiment is to measure the feasibility of the solution, its *makespan* and the time needed for the computation. The environment used for the experiments is rather standard and simple (Intel Duo 2,8Ghz, 1GB RAM, Linux FC5).

To do the experiment, we generated 18 different workflows which:

- Have different topologies.

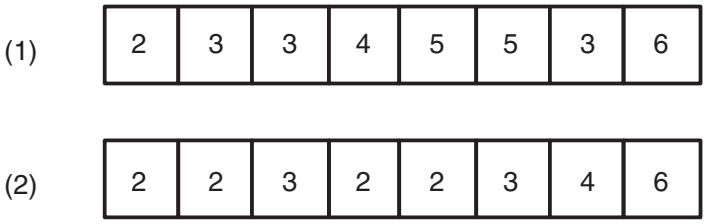


Fig. 11. Newly created configurations

- Have a different number of sub-jobs with the number of sub-jobs being in the range from 7 to 32.
- Have different sub-job specifications. Without loss of generality, we assume that each sub-job has the same CPU performance requirement.
- Have different amounts of data transfer.

The runtime of each sub-job in each type of RMS is assigned by using Formula 4.

$$rt_j = \frac{rt_i}{\frac{pk_i + (pk_j - pk_i) * k}{pk_i}} \tag{4}$$

With pk_i, pk_j is the performance of a CPU in RMS r_i, r_j respectively and rt_i is the estimated runtime of the sub-job with the resource configuration of RMS r_i. k is the speed up control factor. Within the performance experiment, in each workflow, 60% of the number of sub-jobs have $k = 0.5$, 30% of the number of sub-jobs have $k = 0.25$, and 10% of the number of sub-jobs have $k = 0$.

The complexity of the workflow depends on the number of sub-job in the workflow. In the experiment, we stop at 32 sub-jobs for a workflow because it is much greater than the size of the recognized workflows. As far as we know, with our model of parallel task sub-job, most existing scientific workflows as described by Ludtke et al. Ludtke et al. (1999), Berriman et al. Berriman et al. (2003) and Lovas et al. Lovas et al. (2004) include just 10 to 20 sub-jobs.

As the difference in the static factors of an RMS such as OS, CPU speed and so on can be easily filtered by SQL query, we use 20 RMSs with the resource configuration equal to or even better than the requirement of sub-jobs. Those RMSs have already had some initial workload in their resource reservation and bandwidth reservation profiles. In the experiment, 30% of the number of RMS have CPU performance equals to the requirement, 60% of the number of RMS have CPU performance which is 100% more powerful than requirement, 10% of the number of RMS have CPU performance which is 200% more powerful than requirement.

We created 20 RMSs in the experiment because it closely parallels the real situation in Grid Computing. In theory, the number of sites joining a Grid can be very large. However, in reality, this number is not so great. The number of sites providing commercial service is even smaller. For example, the Distributed European Infrastructure for Supercomputing Applications (DEISA) has only 11 sites. More details about the description of resource configurations and workload configurations can be seen at the address: http://it.i-u.de/schools/altmann/DangMinh/desc_expe2.txt.

Sjs	0	100	200	400	600	800	1000
Simple level experiment							
7	56	52	52	52	52	52	52
8	187	55	55	55	55	55	55
9	93	71	64	64	64	64	64
10	81	56	56	56	56	56	56
11	229	65	65	65	65	65	65
12	88	88	88	88	88	88	88
13	149	52	52	52	52	52	52
Intermediate level experiment							
14	218	149	149	149	149	115	115
15	243	185	185	185	185	185	185
16	196	180	180	180	180	180	170
17	73	49	49	49	49	49	49
18	269	207	144	144	144	144	144
19	216	87	87	86	86	86	86
20	248	151	151	151	151	151	151
Advance level experiment							
21	76	37	37	37	37	37	37
25	289	262	217	217	214	205	204
28	276	229	201	76	76	76	76
32	250	250	250	250	250	205	205

Table 4. w-GA convergent experiment results

5.1 Time to convergence

To study the convergence of the w-GA algorithm, we do three levels of experiments according to the size of the workflow. At each level, we use the w-GA to map workflows to the RMSs. The maximum number of generations is 1000. The best found *makespan* is recorded at 0, 100, 200, 400, 600, 800 and 1000 generations. The result is presented in Table 4.

From the data in the Table 4, we see a trend that the w-GA algorithm needs more generations to convergence when the size of the workflow increases.

At the simple level experiment, we map workflow having from 7 to 13 sub-jobs to the RMSs. From this data, we can see that the w-GA converges to the same value after fewer than 200 generations in most case.

At the intermediate level of the experiment where we map a workflow having from 14 to 20 sub-jobs to the RMSs, the situation is slightly different than the simple level. In addition to many cases showing that the w-GA converges to the same value after fewer than 200 generations, there are some cases where the algorithm found a better solution after 600 or 800 generations.

When the size of the workflow increases from 21 to 32 sub-jobs as in the advanced level experiment, converging after fewer than 200 generations happens in only one case. In other cases, the w-GA needs from 400 to more than 800 generations.

5.2 Performance comparison

We have not noticed a similar resource model or workflow model as stated in Section 2. To do the performance evaluation, in the previous work we implemented the w-DCP, Grasp, minmin, maxmin, and suffer algorithms to our problem Quan (2007). The extensive experiment result is shown in Figure 12.

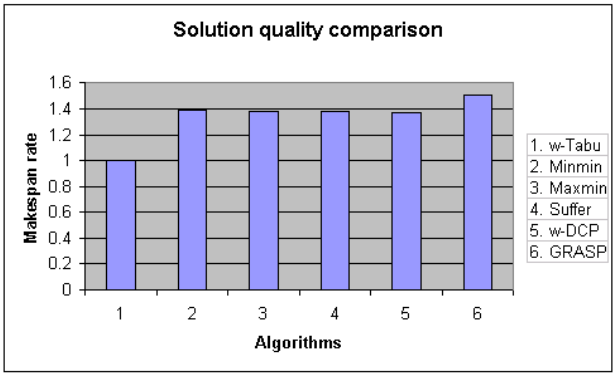


Fig. 12. Overall performance comparison among w-Tabu and other algorithms
Quan (2007)

The experiment result in Figure 12 shows that the w-Tabu algorithm has the highest performance. For that reason, we only need to consider the w-Tabu algorithm in this work. To compare the performance of the w-GA algorithm with other algorithms, we map 18 workflows to RMSs using the w-GA, the w-Tabu, and the n-GA algorithms. Similar to the experiment studying the convergence of the w-GA algorithm, this experiment is also divided into three levels according to the size of the workflow. With the n-GA algorithm, we run it with 1000 generations. With w-GA algorithm, we run it with 120 generations and 1000 generations and thus we have the w-GA1000 algorithm and the w-GA120 algorithm respectively. The purpose of running the w-GA at 1000 generations is for theoretical purpose. We want to see the limit performance of w-GA and n-GA within a long enough period. Thus, with the theoretical aspect, we compare the performance of the w-GA1000, the w-Tabu and the n-GA1000 algorithms. The purpose of running w-GA at 120 generations is for practical purposes. We want to compare the performance of the w-Tabu algorithm and the w-GA algorithm in the same runtime. With each mapping instance, the *makespan* of the solution and the runtime of the algorithm are recorded. The experiment results are presented in Table 5.

In three levels of the experiments, we can see the domination of the w-GA1000 algorithm. In the whole experiment, w-GA1000 found 14 better and 3 worse solutions than did the n-GA1000 algorithm and the w-Tabu algorithm. The overall performance comparison in average relative value is presented in Figure 13. From this Figure, we can see that the w-GA1000 is about 21% better than the w-Tabu and the n-GA1000 algorithms. The data in the Table 5 and Figure 13 also show an equal performance between the w-Tabu and the n-GA1000 algorithms.

	w-GA 1000		w-GA 120		w-Tabu		n.GA 1000	
Sjs	Mksp	Rt	Mksp	Rt	Mksp	Rt	Mksp	Rt
Simple level experiment								
7	52	20	52	2	56	2	67	19
8	55	25	55	3	144	2	67	23
9	64	29	71	3	79	1	79	24
10	56	31	56	4	81	3	94	27
11	65	39	65	4	102	4	160	37
12	88	45	88	5	58	2	62	39
13	52	48	52	5	54	3	65	44
Intermediate level experiment								
14	115	40	149	4	154	3	128	37
15	185	43	185	5	81	4	85	40
16	170	47	180	5	195	3	195	44
17	49	54	49	6	54	4	63	49
18	144	52	207	5	171	5	144	48
19	86	51	87	5	201	5	150	47
20	151	60	151	6	193	4	205	57
Advance level experiment								
21	37	74	37	8	37	7	47	70
25	204	59	262	6	195	8	195	55
28	76	108	229	11	86	7	105	105
32	205	111	250	12	250	10	239	106

Table 5. Performance comparison among w-GA and other algorithms

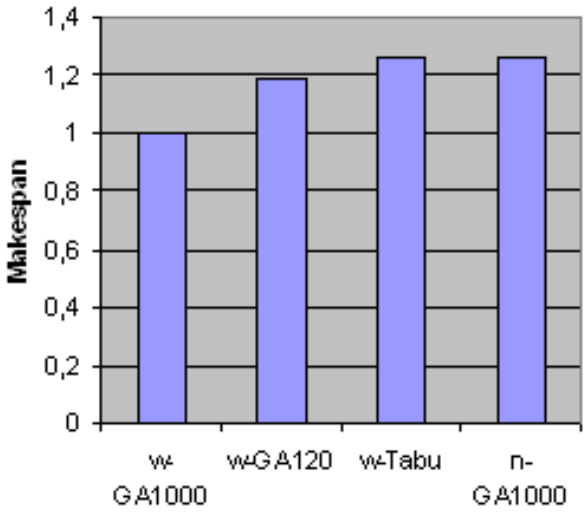


Fig. 13. Overall performance comparison among w-GA and other algorithms

With the runtime aspect, the runtime of the w-GA1000 algorithm is slightly greater than the n-GA1000 algorithm because the w-GA is more complicated than the n-GA. However, the runtime of both the w-GA1000 and the n-GA1000 are much, much longer when compare to the runtime of w-Tabu algorithm. On average, the runtime of the w-GA1000 and the n-GA1000 are 10 times longer than the runtime of the w-Tabu algorithm.

The long runtime of the w-GA1000 and the n-GA1000 is the great disadvantage for them to be employed in the real environment. In practice, thought, the broker scheduling a workflow for 1 or 2 minutes is not acceptable. As the w-Tabu algorithm needs only from 1 to 10 seconds, we run the w-GA algorithm at 120 generations so it has relatively the same runtime as w-Tabu algorithm. As the n-GA algorithm does not have a good performance even at 1000 generations, we will not consider it within the practical framework. In particular, we focus on comparing the performance of the w-GA120 and the w-Tabu algorithm.

From the data in Table 5, we see a trend that the w-GA120 decreases its performance compared to the w-Tabu when the size of the workflow increases.

At the simple level and intermediate level of the experiment, the quality of the w-GA120 is better than the quality of the w-Tabu algorithm. The w-GA algorithm found 3 worse solutions and 11 better solutions than the w-Tabu algorithm.

However, at the advance level experiment, the quality of the w-GA120 is not acceptable. Apart from one equal solution, the w-GA120 found more worse solutions than the w-Tabu algorithm. This is because of the large search space. With a small number of generations, the w-GA cannot find high quality solutions.

6. The combined algorithm

From the experiment results of the w-GA120 and w-Tabu algorithms, we have noted the following observations.

- The w-Tabu algorithm has runtime from 1 to 10 seconds and this range is generally acceptable. Thus, the mapping algorithm could make use of the maximum value of allowed time period, i.e 10 seconds in this case, to find the highest possible quality solution.
- Both the w-GA and the w-Tabu found solutions with great differing quality in some cases. This means in some case the w-GA found a very high quality solution but the w-Tabu found very low quality solutions and vice versa.
- When the size of the workflow is very big and the runtime of the w-GA and the w-Tabu to find out solution also reaches the limit, the quality of the w-GA120 is not as good as the w-Tabu algorithm.

From these observations, we propose an other algorithm combining the w-GA120 and the w-Tabu algorithm. The new algorithm called w-TG is presented in Algorithm 6.

From the experiment data in Table 5, the runtime of the w-TG algorithm is from 4 to 10 seconds. We run the w-GA with 120 generations in all cases for two reasons.

- If the size of the workflow is large, increasing the number of generations will significantly increase the runtime of the algorithm. Thus, this runtime may exceed the acceptable range.

Algorithm 6 w-TG algorithm

```
1: if the size of the workflow <= 20 then
2:     Call w-GA120 to find solution a1
3:     Call w-Tabu to find solution a2
4:     a'' ← better(a1, a2)
5: else
6:     Call w-Tabu to find solution a"
7: end if
8: return a"
```

- If the size of the workflow is small, the algorithm has high probability of convergence within a small number of generations. The data in Table 4 also supports this idea.

To examine the performance of the w-TG algorithm, we do an extensive experiment in order to make a comparison with the w-GA and the w-Tabu algorithm. For this experiment, we keep the topology of 18 workflows as in the experiment in Section 4 but change the configuration of sub-jobs in each workflow. With each topology we created 5 different workflows. Thus, we have a total 90 different workflows.

Those workflows are mapped to the RMSs using the w-GA, the w-Tabu and the w-TG algorithms. The *makespan* of the solution and the runtime of the algorithm are then recorded. From the experiment data, the runtime of all algorithms is from 1 to 12 seconds. The average performance of each algorithm is presented in Figure 14 and Figure 15.

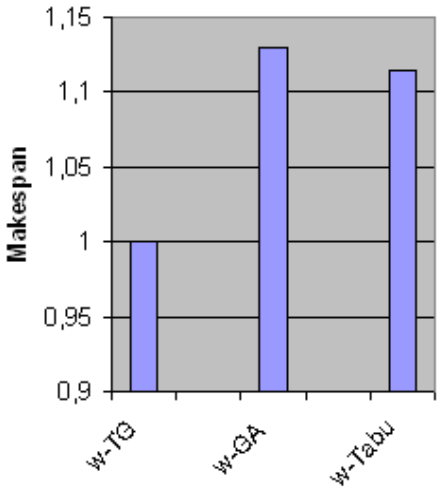


Fig. 14. Performance comparison when the size of each workflow less than or equal to 20

Figure 14 presents the comparison of the average *makespan* in relative value when all the workflows in the experiment have the number of sub-job less than or equal to 20. We want to see the performance of the equal combination part of the w-TG algorithm. As can be seen from Figure 14, the w-TG algorithm has the highest performance. The w-TG algorithm found solutions 11% better than the w-Tabu algorithm and 12% better than the w-GA120 algorithm.

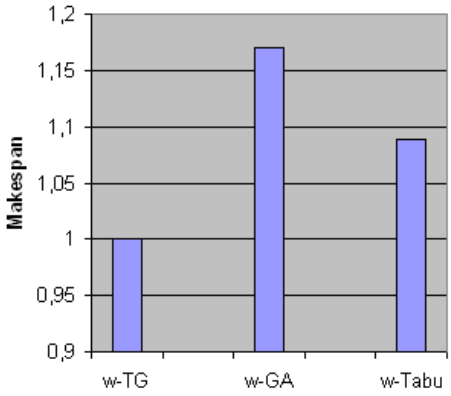


Fig. 15. Performance comparison when the size of each workflow less than or equal to 32

We make the average comparison with all 90 workflows, and the overall result of this is presented in Figure 15. As with the workflow having the number of sub-jobs more than 20, the quality of the w-TG algorithm is equal to the quality of the w-Tabu algorithm. Thus the better rate of the w-TG compared to the w-Tabu is reduced by about 9%. In contrast, as the quality of the w-GA algorithm is not as good with a workflow having the number of sub-jobs more than 20. Thus the worse rate of the w-GA algorithm compared to the w-TG algorithm is an increase to 17%.

We can also see that the performance of the w-GA120 compared to the w-Tabu in this experiment is not as high as in the experiment of Section 5.2. This means that the performance of the w-GA120 fluctuates with different scenarios. However, in any case, the combined algorithm still has good performance.

7. Conclusion

In this book chapter we presented the modified Genetic Algorithm and its combination with the w-Tabu algorithm to form a new algorithm called w-TG to solve the problem of optimizing runtime of the Grid-based workflows within the SLA context. In our work, the distinguishing characteristic is that each sub-job of a workflow can be either a sequential or parallel program. In addition, each grid service can handle many sub-jobs at a time and its resources are reserved. The w-Tabu algorithm creates a set of referent solutions, which distribute widely over the search space, and then searches around those points to find the local minimal solution. We proposed a special genetic algorithm to map workflow to the Grid resources called w-GA. In the w-GA algorithm, we applied many dedicated techniques for workflow within the crossover and mutation operations in order to improve the searching quality. The experiment showed that both the w-GA and the w-Tabu found solutions with great differing quality in some cases. When the size of the workflow is very big and the runtime of the w-GA and the w-Tabu to find out solution also reaches the limit, the quality of the w-GA is not as good as the w-Tabu algorithm. The combined algorithm can fix the disadvantage of the individual algorithms. Our performance evaluation showed that the combined algorithm created solution of equal or better quality than the previous algorithm

and requires the same range of computation time period. The latter is a decisive factor for the applicability of the proposed method in real environments.

8. References

Adam, T. L., Chandy, K. M. and Dickson, J. R., 1974, A comparison of list scheduling for parallel processing systems. *Communication of the ACM*, 17, 685-690.

Ayyub, S. and Abramson, D. (2007) 'GridRod - A Service Oriented Dynamic Runtime Scheduler for Grid Workflows'. *Proceedings of the 21st ACM International Conference on Supercomputing*, pp. 43-52.

Berman *et al*. 2005 'New Grid Scheduling and Rescheduling Methods in the GrADS Project', *International Journal of Parallel Programming*, Vol. 33, pp.209-229.

Berriman, G. B., Good, J. C., Laity, A. C. (2003) 'Montage: a Grid Enabled Image Mosaic Service for the National Virtual Observatory', *ADASS*, Vol. 13, pp.145-167.

P. E. Black, "Algorithms and Theory of Computation Handbook", CRC Press LLC, 1999.

Blythe *et al*. 2005 'Task Scheduling Strategies for Workflow-based Applications in Grids', *Proceeding of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005)*, pp.759-767.

Casanova, H., Legrand, A., Zagorodnov, D. and Berman, F. 2000 'Heuristics for Scheduling Parameter Sweep applications in Grid environments', *Proceeding of the 9th HeterogeneousComputing workshop (HCW'2000)*, pp.292–300.

Coffman, E. G., 1976, Computer and Job-Shop Scheduling Theory. John Wiley and Sons, Inc., New York, NY.

Colin, J. Y. and Chretienne, P., 1991, Scheduling with small computation delays and task duplication. *Operation Research*, 39, 680-684.

CondorVersion 6.4.7 Manual. www.cs.wisc.edu/condor/manual/v6.4 [10 December 2004].

Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M., Vahi, K. and Livny, M. (2004) 'Pegasus : Mapping Scientific Workflows onto the Grid', *Proceedings of the 2nd European Across Grids Conference*, pp.11-20.

Duan, R., Prodan, R., Fahringer, T. (2006) 'Run-time Optimization for Grid Workflow Applications', *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing (Grid'06)*, pp. 33-40.

Elmagarmid, A.K. (1992) *Database Transaction Models for Advanced Applications*, Morgan Kaufmann.

Gary, M. R. and Johnson, D. S., 1979, Computers and Intractability: A Guide to the theory of NP-Completeness. W. H. Freeman and Co.

Georgakopoulos, D., Hornick, M., and Sheth, A. (1995) 'An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure', *Distributed and Parallel Databases*, Vol. 3, No. 2, pp.119-153.

Gerasoulis, A. and Yang, T., 1992, A comparison of clustering heuristics for scheduling DAG's on multiprocessors. *J. Parallel and Distributed Computing*, 16, 276-291.

Hou, E. S. H., Ansari, N., and Ren, H., 1994, A genetic algorithm for multiprocessor scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 5, 113-120.

Hovestadt, M. (2003) 'Scheduling in HPC Resource Management Systems:Queuing vs. Planning', *Proceedings of the 9th Workshop on JSSPP at GGF8*, LNCS, pp.1-20.

Hsu, M. (ed.) (1993) *Special Issue on Workflow and Extended Transaction Systems*, IEEE Data Engineering, Vol. 16, No. 2.

Kohler, W. H. and Steiglitz, K., 1974, Characterization and theoretical comparison of branch-and-bound algorithms for permutation problems. *Journal of ACM*, 21, 140-156.

Kruatrachue, B. and Lewis, T. G., 1987, Duplication Scheduling Heuristics (DSH): A New Precedence Task Scheduler for Parallel Processor Systems. Oregon State University, Corvallis, OR.

Kruatrachue , B., and Lewis, T., 1988, Grain size determination for parallel processing. *IEEE Software*, 5, 23-32.

Kwok Y. K. and Ahmad, I., 1999, Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Computing Surveys (CSUR)*, 31, 406-471.

Lovas, R., Dózsa, G., Kacsuk, P., Podhorszki, N., Drótos, D. (2004) 'Workflow Support for Complex Grid Applications: Integrated and Portal Solutions', *Proceedings of 2nd European Across Grids Conference*, pp.129-138.

Ludtke, S., Baldwin, P. and Chiu, W. (1999) 'EMAN: Semiautomated Software for High-Resolution Single-Particle Reconstructio' , *Journal of Structure Biology*, Vol. 128, pp. 146–157.

Ma, T. and Buyya, R. 2005 'Critical-Path and Priority based Algorithms for Scheduling Workflows with Parameter Sweep Tasks on Global Grids', *Proceeding of the 17th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2005), IEEE CS Press*, pp.251-258.

Mello, R. F., Filho J. A. A., Senger, L. J., Yang, L. T. (2007) 'RouteGA: A Grid Load Balancing Algorithm with Genetic Support', *Proceedings of the 21st International Conference on Advanced Networking and Applications, (AINA 2007), IEEE CS Press*, pp.885-892.

Quan, D.M., Kao, O. (2005) 'On Architecture for an SLA-aware Job Flows in Grid Environments', *Journal of Interconnection Networks*, Vol. 6, No. 3, pp.245-264.

Quan, D.M., Hsu, D. F. (2006) 'Network based resource allocation within SLA context', *Proceedings of the GCC2006*, pp. 274-280.

Quan, D.M., Altmann, J. (2007) 'Business Model and the Policy of Mapping Light Communication Grid-Based Workflow Within the SLA Context', *Proceedings of the International Conference of High Performance Computing and Communication (HPCC07)*, pp.285-295.

Quan, D.M. (2007) 'Error recovery mechanism for grid-based workflow within SLA context', *Int. J. High Performance Computing and Networking*, Vol. 5, No. 1/2, pp.110-121.

Quan, D.M., Altmann, J. (2007) 'Mapping of SLA-based Workflows with light Communication onto Grid Resources', *Proceedings of the 4th International Conference on Grid Service Engineering and Management (GSEM 2007)*, pp.135-145

Quan, D.M., Altmann, J. (2007) 'Mapping a group of jobs in the error recovery of the Grid-based workflow within SLA context', *Proceedings of the 21st International Conference on Advanced Networking and Applications, (AINA 2007), IEEE CS Press*, pp.986-993.

Rewini , H. E. and Lewis, T. G., 1990, Scheduling parallel program tasks onto arbitrary target machines. *Journal of Parallel and Distributed Computing*, 9, 138-153.

Shahid, A., Muhammed, S. T. and Sadiq, M., 1994, GSA: scheduling and allocation using genetic algorithm. Paper presented at the Conference on European Design Automation, Paris, France, 19-23 September.

Sahai, A., Graupner, S., Machiraju, V. and Moorsel, A. 2003 'Specifying and Monitoring Guarantees in Commercial Grids through SLA', *Proceeding of the 3rd IEEE/ACM CCGrid2003*, pp.292–300.

Sarkar, V., 1989, Partitioning and Scheduling Parallel Programs for Multiprocessors. MIT Press, Cambridge, MA.

Sih, G. C., and Lee, E. A., 1993, Declustering: a new multiprocessor scheduling technique. *IEEE Transactions on Parallel and Distributed Systems*, 4, 625-637.

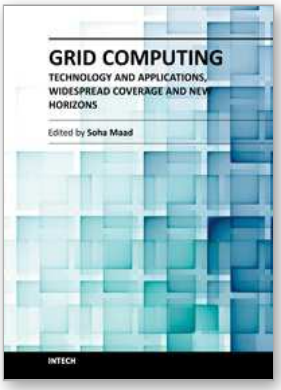
Singh, M. P. and Vouk, M. A. (1997) *Scientific Workflows: Scientific Computing Meets Transactional Workflows*, http://www.csc.ncsu.edu/faculty/mpsingh/papers/databases/workflows/sciworkflows.html

Spooner, D. P., Jarvis, S. A., Cao, J., Saini, S. and Nudd, G. R. (2003) 'Local Grid Scheduling Techniques Using Performance Prediction', ÎEEE Proceedings - Computers and Digital Techniques, pp.87–96.

Yu, J., Buyya R. (2005) 'A taxonomy of scientific workflow systems for grid computing', *SIGMOD Record*, Vol. 34, No. 3, pp.44-49.

Fischer, L. *Workflow Handbook 2004*, Future Strategies Inc., Lighthouse Point, FL, USA.

Wolski, R. (2003) 'Experiences with Predicting Resource Performance On-line in Computational Grid Settings', *ACM SIGMETRICS Performance Evaluation Review*, Vol. 30, No. 4, pp.41-49.



Grid Computing - Technology and Applications, Widespread Coverage and New Horizons

Edited by Dr. Soha Maad

ISBN 978-953-51-0604-3
Hard cover, 354 pages
Publisher InTech
Published online 16, May, 2012
Published in print edition May, 2012

Grid research, rooted in distributed and high performance computing, started in mid-to-late 1990s. Soon afterwards, national and international research and development authorities realized the importance of the Grid and gave it a primary position on their research and development agenda. The Grid evolved from tackling data and compute-intensive problems, to addressing global-scale scientific projects, connecting businesses across the supply chain, and becoming a World Wide Grid integrated in our daily routine activities. This book tells the story of great potential, continued strength, and widespread international penetration of Grid computing. It overviews latest advances in the field and traces the evolution of selected Grid applications. The book highlights the international widespread coverage and unveils the future potential of the Grid.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821