

# Network-based resource allocation for Grid Computing within an SLA context

Dang Minh Quan  
University of Paderborn  
Paderborn Center for Parallel Computing  
33102 Paderborn, Germany  
quandm@upb.de

D. Frank Hsu  
Fordham University  
Department of Computer and Information Science  
New York, NY 10023, USA,  
hsu@cis.fordham.edu

## Abstract

*Service Level Agreements (SLAs) are currently one of the major research topics in Grid Computing. Among many system components for supporting of SLA-aware Grid jobs, the SLA mapping mechanism receives important positions. It is responsible for assigning sub-jobs of the workflow to Grid resource in a way that meet the user's deadline and as cheap as possible. With the distinguished workload and resource characteristics, mapping a heavy communication workflow within SLA context defines an unfamiliar problem and need a specific method to be solved. This paper presents the mapping algorithm, which can cope with the problem. Performance measurements deliver evaluation results on the quality and efficiency of the method.*

## 1. Introduction

Mapping and running jobs on suitable resources are the core tasks in Grid Computing. With the case of Grid-based workflows, where a single job is divided into several sub-jobs, the majority of efforts for this issue such as [5, 15, 7] work to map a workflow on the Grid resources in best effort manner. In the SLA (Service Level Agreement) [14] context, where resources are reserved to ensure the Quality of Service (QoS), mapping a workflow on the Grid resources requires different mechanism. The literature recorded some proposed solutions for this problem such as [9, 17, 3]. Most of the proposed mechanisms suppose a workflow including many sub-jobs, which are sequent programs and a Grid service having ability to handle one sub-job at a time. This is not sufficient enough as sub-jobs in many existed workflows [8, 1, 13] are parallel programs and many High Performance Computing Center (HPCC) provide computing service under single Grid service [4]. It is obvious that a HPCC can handle many sub-jobs, which can be either sequent programs or parallel programs, at a time. Moreover, all of them did not consider the case of having heavy communication

among sub-jobs in the workflow. This paper, which is a continuous work in a series of efforts supporting SLA for the Grid-based workflow [11, 10, 12], will present a mechanism to handle all stated drawbacks.

### 1.1 Workflow model

Like many popular system handling Grid-based workflow [5, 15, 7], we also suppose Directed Acyclic Graph (DAG) form of the workflow. User describes specification about required resources to run sub-job, data transfer among sub-jobs, the estimated runtime of sub-jobs and impose the expected runtime of the whole workflow. User wants the system to finish running the whole workflow in time. In the scope of this paper, the time is computed in slot. Each slot equals with a specific period of real time. Figure 1 presents a concrete example Grid workflow. Each sub-job of the workflow has different resource requirements as described in table 2.

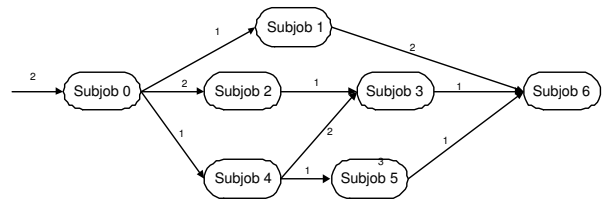


Figure 1. A sample workflow

It is noted that a sub-job of the workflow can be either sequent program or parallel program and the data to be transferred among sub-jobs is very large, usually in GB scale. The case of light communication among sub-jobs of the workflow was handled in [10]

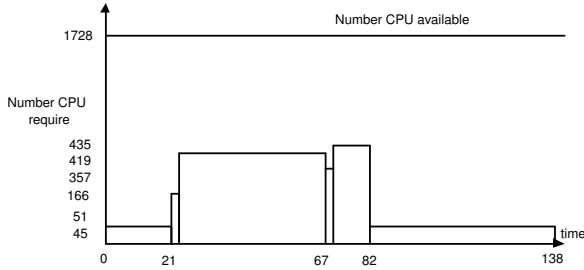
### 1.2 Grid service model

The computational Grid includes many High Performance Computing Centers (HPCCs). We believe that only

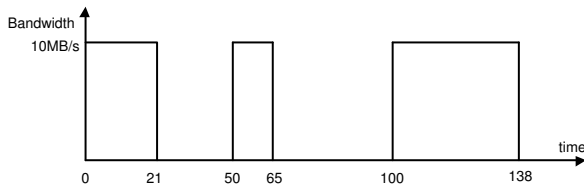
Sj_ID	CPU	Storage	exp	runtime
0	18	59	1	6
1	16	130	3	8
2	20	142	4	5
3	22	113	4	8
4	18	174	2	9
5	20	97	3	14
6	16	118	1	4

**Table 1. Resource requirements for subjobs**

HPCCs have enough conditions to support SLA for sub-jobs of the workflow. The resources in each HPCC are managed by the software called local Resource Management System (RMS). In this paper, the acronym RMS is used to represent the HPCC as well as the Grid service of that HPCC. Each RMS has its own resource configuration and this configuration is usually different from each other. Those differences include number of CPU, number of memory, storage capacity, software, expert, service price, etc. To ensure that the sub-job can be executed within a dedicated time period, the RMS must supports advance resource reservation, for example CCS [6]. Figure 9 depicts a sample CPU reservation profile in such RMS. Queuing-based RMSs are not suitable for our requirement, as no information about the starting time is provided. In our system, we reserve three main types of resource: CPUs, storages and experts. An extension to other devices, software licenses, and other related resources is straightforward.



**Figure 2. A sample CPU reservation profile of a local RMS**



**Figure 3. A sample bandwidth reservation profile of a link between two local RMSs**

If two sequent sub-jobs are executed in the same RMS, it is not necessary to do data transfer task and the time used for this work equal 0. Otherwise, data transfer task between them must be performed. To make sure that a specific amount of data will be transferred within a specific period of time, the bandwidth must also be reserved. Unfortunately, up to now, there is no mechanism responsible for that task in the worldwide network. Here, to overcome that elimination, we use central broker mechanism. The link bandwidth between two local RMSs is determined as the average bandwidth between two sites in the network, which has different value with each different couple RMSs. Whenever having a data transfer task on a link, the SLA broker will determine which time slot is available for that task. During that specified period, the task can use the whole bandwidth and other tasks must wait. Using this principle, the bandwidth reservation profile of a link will look similar to the one, which is depicted in Figure 3. A more correctly model with bandwidth estimation [16] can be used to determine the bandwidth within a specific time period instead of the average value. In both cases, the main mechanism is unchanged.

### 1.3 Mapping mechanism requirement

The formal specification of the described problem includes following elements:

- 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 current resource and deadline requirements.
- Let  $E$  be the set of data transfer 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, a feasible and possibly optimal solution is sought, which allows the most efficient mapping of the workflow in a Grid environment with respect to the given global deadline. The required solution is a set defined as Formula 1.

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

If the solution does not have start\_slot for each  $s_i$ , it become a configuration as defined in Formula 2.

$$a = \{(s_i, r_j | s_i \in S, r_j \in K_i)\} \quad (2)$$

A feasible solution must satisfy following conditions:

- **Criteria1:** All  $K_i \neq \emptyset$ . There is at least one RMS in the candidate set of each sub-job.
- **Criteria2:** The total runtime period of the workflow must be within the expected period given by user.
- **Criteria3:** The dependencies of the sub-jobs are resolved and the execution order remains unchanged.
- **Criteria4:** Each RMS provides a profile of currently available resources and can run many sub-jobs of a single flow both sequentially and 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.
- **Criteria5:** The data transmission task  $e_{ki}$  from sub-job  $s_k$  to sub-job  $s_i$  must not overlap other reserved data transmission task on the link between RMS running sub-job  $s_k$  to RMS running sub-job  $s_i$ .  $e_{ki} \in E$ .

In the next phase the feasible solution with the lowest cost is sought. The cost  $C$  of a Grid workflow is defined in formula 3, 4, 5. It is a sum of four factors: money for using CPU, money for using storage, cost of using experts knowledge and finally money for transferring data between the involved resources.

$$C_1 = \sum_{i=1}^n s_i \cdot r_t * (s_i \cdot n_c * r_j \cdot p_c + s_i \cdot n_s * r_j \cdot p_s + s_i \cdot n_e * r_j \cdot p_e) \quad (3)$$

$$C_2 = \sum e_{ki} \cdot n_d * r_j \cdot p_d \quad (4)$$

$$C = C_1 + C_2 \quad (5)$$

with  $s_i \cdot r_t, s_i \cdot n_c, s_i \cdot n_s, s_i \cdot n_e$  are the runtime, number CPU, number storage, number expert of sub-job  $s_i$  respectively.  $r_j \cdot p_c, r_j \cdot p_s, r_j \cdot p_e, r_j \cdot p_d$  are the price of using CPU, storage, expert, data transmission of RMS  $r_j$  respectively.  $e_{ki} \cdot n_d$  is the number of data to be transferred from sub-job  $s_k$  to sub-job  $s_i$ .

If two sequent subjobs run on the same RMS, the cost of transferring data from the previous subjob to the later subjob is neglected. It can be shown easily that the optimal mapping of the workflow to Grid RMS with cost optimizing is a NP hard problem.

## 2 Related work

In two separated works [17, 3], Zeng et al and Iwona et al built systems to support QoS features for Grid-based workflow. In their work, a workflow includes many sub-jobs, which are sequent programs, and a Grid service has ability to handle one sub-job at a time. To map the workflow on to the Grid services, they used Integer Programming method. Applying Integer Programming to our problem faces many difficulties. The first is the flexibility in runtime of the data transfer task. The time to complete data transfer task depends on the bandwidth and the reservation profile of the link, which varies from link to link. The variety in completion time of data transfer task makes the constraints presentation very complicated. The second is that a RMS can handle many parallel programs at a time. Thus, presenting the constraints of profile resource requirement and profile of resource available in Integer Programming is very difficult to perform.

With the same resource reservation and workflow model, we proposed an algorithm which mapping a light communication workflow to Grid resources in [10]. The proposed algorithm uses Tabu search to find the best possible assignment of sub-jobs to resources. In order to shorten the computation time caused by the high number of resource profiles to be analyzed and by the flexibility while determining start and end times for the sub-jobs, several techniques for reducing the search space are introduced. However, these techniques can not be applied to solve the problem in this paper because the bandwidth reservation model is not considered in [10].

Metaheuristics such as GA, Simulated Annealing [2], etc were proved to be very effective in mapping, scheduling problems. McGough et al also use them in their system [9]. However, in our problem, with the appearance of resource profiles, the evaluation at each step of the search is very hard. If the problem is big with highly flexible variable, the classical searching algorithm need very long time to find a good solution. In the scope of this paper, we apply several metaheuristics to our problem as means of comparing.

## 3 Planning algorithm for heavy communication workflows

Each sub-job has different resource requirements about type of RMS, type of CPU, etc. There are a lot of RMSs with different resource configuration. The initial action is finding among those heterogeneous RMSs the suitable RMSs, which can meet the requirement of the sub-job. Data of each resource parameter of RMS is represented by number value and is stored in separate column in the database table. For example, with the parameter Operating System, OSs such as Linux, Sun, Window, Unix are represented by

value number 1, 2, 3, 4 respectively. The co-relative resource requirement parameter of a sub-job is also represented by number value in the same manner. Thus, 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 Criteria 1. Suppose that each sub-job has  $m$  RMSs in the candidate list, we could have  $m^n$  configuration. For present purpose, we use workflow including 7 sub-jobs as presented in Figure 1 and each sub-job having 3RMSs in the candidate list. The algorithm to mapping DAG to resources is described as following.

### 3.1 Algorithm A - Constructing initial configurations

The purpose of this algorithm is creating a set of initial configurations, which distributes widely over the search space.

**Step 0:** With each sub-job  $s_i$ , we sort the RMSs in the candidate set  $K_i$  according to the cost they need to run  $s_i$ . The cost is computed according to Formula 5. The configuration space of the sample now can be presented in Figure 4 and Table 2. In Figure 4, the RMSs lying along the axis of each sub-job have increasing cost in the direction from inside to outside. The line connecting each point in every sub-job axis will form a configuration. Figure 4 presents 3 configurations with increasing index in the direction from inside to outside. Figure 4 also presents the cost distribution of the configuration space according to Formula 5. The configuration in outer layer has bigger cost than the configuration in the inner layer. The cost of the configuration lying between two layers is bigger than the cost of the inner layer and smaller than the cost of the outer layer.

**Step 1:** We pick the first configuration as the first layer in the configuration space. The determined configuration can be presented as a vector. The index of the vector represents the sub-job, value of the element represents the RMS. The first configuration for our example is presented in Figure 5. Although the first configuration has minimal cost according to Formula 5, we cannot sure that this is the optimal solution. The real cost of a configuration must consider the neglected cost of data transmission when two sequent sub-jobs are in the same RMS.

**Step 2:** We construct the other configurations by doing a process similar to the one described in Figure 6. The second solution is the second layer of the configuration space. Then we create a solution having cost located between layer 1 and layer 2 by combining the first and the second configuration. To do this, we take the  $p$  first elements from the first vector configuration and then  $p$  second elements from the second vector configuration and repeat until having  $n$  elements to

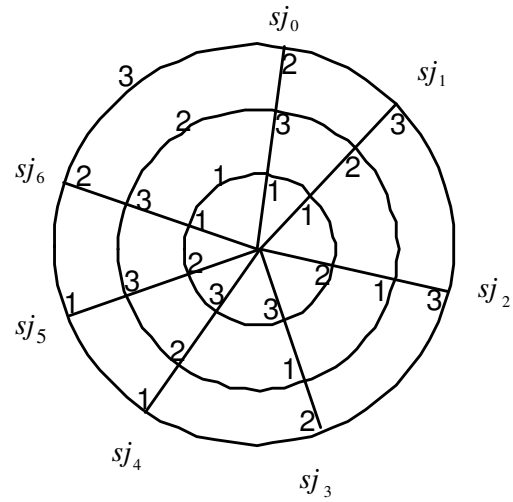


Figure 4. The configuration space according to cost distribution

Sj_ID	RMS	RMS	RMS
sj0	R1	R3	R2
sj1	R1	R2	R3
sj2	R2	R1	R3
sj3	R3	R1	R2
sj4	R3	R2	R1
sj5	R2	R3	R1
sj6	R1	R3	R2

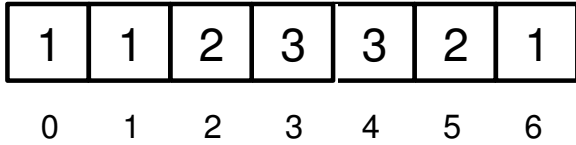
Table 2. RMS candidate for each sub-job in cost order

form the third one. Thus, we get  $(n/2)$  elements from the first vector configuration and  $(n/2)$  other elements from the second one. Combining by this way will ensure the target configuration having maximal difference in cost according to Formula 5 comparing to the source configurations. The process continues until we reach the final layer. Thus, we have totally  $2*(m-1)$  configurations. With this method we can ensure that the set of initial configurations is distributed over the search space according to cost criteria.

**Step 3:** We check the Criteria 2 of all  $2*m-1$  configurations. To verify Criteria 2, we have to determine the timetable for all sub-jobs of the workflow. In general, to ensure the integrity of the workflow, sub-jobs in the workflow must be assigned based on the sequence of the data processing. This principle will ensure Criteria 3. However, that principle does not cover the case of a set of sub-jobs, which have the same priority in data sequence and do not depend on each other. To examine the problem, we determine the earliest and the latest start time of each sub-jobs of the workflow in ideal condition. The time period to do data

Sub-job	Earliest start	Latest start
0	0	0
1	7	22
2	8	17
3	18	23
4	7	7
5	17	17
6	32	32

**Table 3. Valid start time for sub-jobs of workflow in Figure 1**

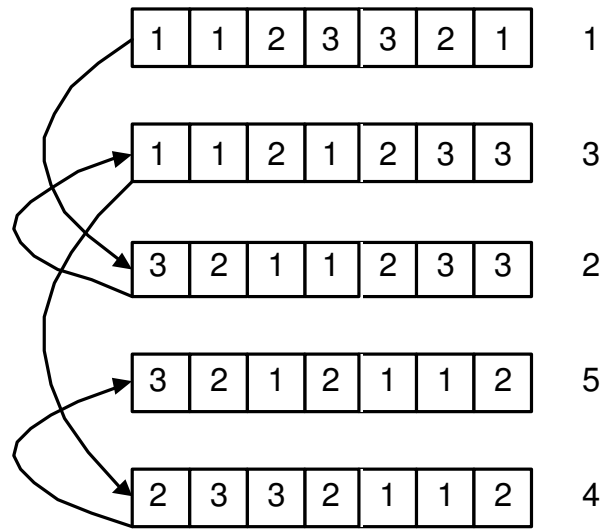


**Figure 5. The first selection configuration of the sample**

transmission among sub-jobs is computed by dividing the amount of data to a fix bandwidth. The earliest and latest start, stop time for each sub-job and data transfer depends only to the workflow topology and the runtime of sub-jobs but not the resources context. Those parameters can be determined by using conventional graph algorithms. A sample of those data for the workflow in Figure 1, in which the number above each link represents number of time slots to data transmission, is presented in Table 3.

The ability of finding a suitable resource slot to run a sub-job depends on number of resource free during the valid running period. From the graph, we can see sub-job 1 and sub-job 4 having the same priority in data sequence. However, from the data in table 3, sub-job 1 can start at max time slot 22 while sub-job 4 can start at max time slot 7 without affecting the finished time of workflow. Suppose that two sub-jobs are mapped to run in the same RMS and the RMS can run one sub-job at a time. If sub-job 4 is assigned first at time slot 7, sub-job 1 will be run from time slot 16 thus the workflow will not be late. If sub-job 1 is assigned first, in the worse case at time slot 22, sub-job 4 can be run at time slot 30 and the workflow will late 23 time slots. Here we can see, the latest time factor is the main parameter to evaluate the full affection of the sequence assignment decision. It can be seen through the affection, mapping sub-job having smaller latest start time first will make the latency smaller. Thus, the latest start time value determined as above can be used to determine the assign sequence. Sub-job having smaller latest start time will be assigned earlier.

For each sub-job of the workflow in the assigning sequence, first we find all the runtime period of data transfer-



**Figure 6. Procedure to create the set of initial configurations**

ring task from previous sub-jobs to current sub-job. This period must be later than the finish time of the source sub-job. Note that with each different links the transfer time is different because of different bandwidth. Then we determine the runtime period of the sub-job itself. This period must be later than the latest finish time of previous related data transfer task. These actions will ensure Criteria 4 and Criteria 5.

If some of them does not satisfy the Criteria 2 requirement, we construct more to have enough  $2*m-1$  configurations. To do the construction, we change the value of  $p$  parameter in the range from 1 to  $(n/2)$  in step 3 to create the new configuration.

After this phase we have set  $C_0$  including maximum  $(2m-1)$  valid configurations.

### 3.2 Algorithm B - Constructing the neighborhood configurations and optimizing

**Step 0:** Take  $C_0$ , where  $|C_0|=2m-1$ .

With a configuration  $a \in C_0$ ,  $a=a_1a_2...a_n|\forall a_i \in K_i$ , we generate  $n*(m-1)$  configurations  $a'$  as in Figure 7. We change the value of  $x_i$  to every value in the candidate list which is different from the present value. With each change, we have a new configuration. After that we have set  $A$ ,  $|A|=n*(m-1)$ . Detail for the case of our example is in Figure 8.

**Step 1:**  $\forall a \in A$ , calculate  $cost(a)$  and  $timetable(a)$ , picks  $a^*$  with the smallest  $cost(a^*)$  and satisfy Criteria 2, put  $a^*$  to set  $C_1$ .

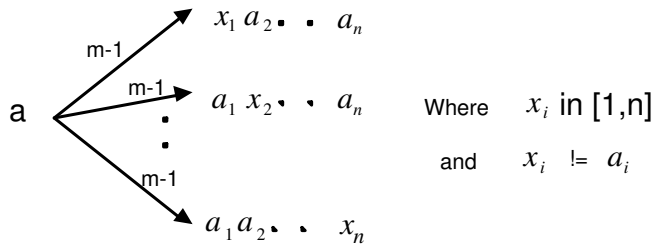


Figure 7. Neighborhood structure of a configuration

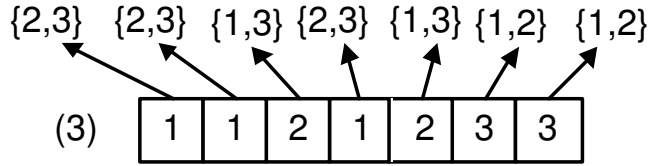


Figure 8. Sample neighborhood structure of configuration 3

- Step 2:** Repeat step 1 and step 2 with all  $a \in C_0$  to form  $C_1$ .
- Step 3:** Repeat step 1 to 3 until  $C_t = C_{t-1}$ .
- Step 4:**  $C_t = C^*$ . Pick the best configuration of  $C^*$ .

### 3.3 Architecture of the frame work

The overall procedure described above is presented in Figure 9.

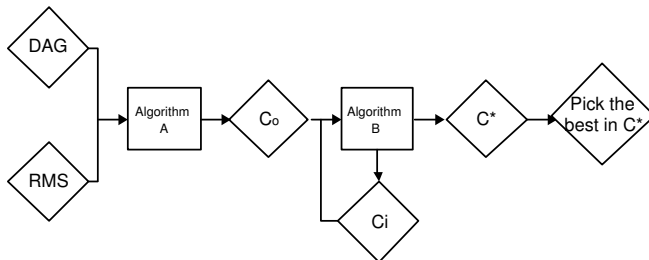


Figure 9. The architecture of the frame work

## 4 Performance evaluation

Performance experiment is done with simulation to check for the quality of the mapping algorithms. The hardware and software used in the experiments is rather standard and simple (Pentium 4 2,8Ghz, 2GB RAM, Linux Redhat 9.0, MySQL). The whole simulation program is implemented in C/C++. With each individual algorithm, we

generated several scenarios with different workflow configurations and different RMS configurations to suit with the ability of the comparing algorithms. The goal of the experiment is to measure the feasibility, the quality of the solution and the time needed for the computation. To do the experiment, 18 workflows with different topologies, number of sub-jobs, sub-job specifications, amount of data transferring were generated. Those workflows will be mapped to 20 RMSs with different resource configurations and different resource reservation contexts by 6 algorithms H-Map, Simulated Annealing (SA), Guided Local Search (GLS), Iterated Local Search (ILS), Genetic Algorithm (GA), Estimation of Distribution Algorithm (EDA) [2]. The final result of the experiment is presented in table 4 and 5. Column Sjs (Sub-jobs) presents the number of sub-jobs of the workflows used in the experiment. The cost and runtime of solutions generated by each algorithm correlative with each workflow is recorded in column Rt (Runtime) and Cost respectively.

	H-Map		SA		ILS	
Sjs	Rt	Cost	Rt	Cost	Rt	Cost
Simple level experiment						
7	2	632.58	105	632.58	21	632.58
8	7	756.90	84	756.90	37	756.96
9	8	780.34	114	780.42	54	780.42
10	13	836.18	78	836.18	81	836.18
11	15	892.02	105	892.02	114	892.02
12	25	947.86	90	948.10	147	948.10
14	31	1003.7	78	1003.99	201	1003.7
Intermediate level experiment						
15	33	1059.54	121	1059.89	250	1059.89
16	29	1183.92	130	1184.21	307	1184.21
17	42	1308.3	146	1332.53	398	1308.53
18	51	1364.14	124	1376.42	502	1364.14
19	41	1488.52	184	1551.74	462	1521.74
20	71	1511.9	174	1512.26	620	1512.26
Advance level experiment						
25	47	1567.74	162	1631.15	815	1567.74
28	43	1591.12	161	1675.67	876	1591.12
32	92	1785.64	180	1871.81	1394	1840.31
34	128	1888.24	197	1960.87	1695	1892.27
35	160	2216.22	272	2276.33	2046	2283.67

Table 4. Performance evaluation result 1

The experiment is divided into 3 levels. In the simple level, 7 workflows having number of sub-jobs in the range 7 - 14 are mapped to 20 RMSs. The result shows that the solutions created by different methods converge to one value with each workflow.

In the intermediate level experiment, we do mapping 6 workflows having number of sub-jobs in the range from 15

Sjs	GLS		GA		EDA	
	Rt	Cost	Rt	Cost	Rt	Cost
Simple level experiment						
7	14	632.58	25	632.58	62	632.58
8	19	756.90	22	756.96	88	756.90
9	29	780.34	27	780.34	126	780.34
10	46	836.18	28	836.18	178	836.18
11	59	892.02	29	892.21	241	892.02
12	80	948.10	36	1005.27	390	947.86
14	98	1003.99	36	1075.19	462	1003.7
Intermediate level experiment						
15	127	1059.89	32	1059.89	558	1059.89
16	167	1184.21	44	1248.86	659	1183.92
17	187	1308.53	47	1383.53	680	1308.53
18	222	1377.63	52	1440.81	956	1364.42
19	303	1501.95	51	1569.39	854	1536.74
20	354	1566.09	56	1620.17	1136	1512.26
Advance level experiment						
25	392	1568.15	56	1663.81	1255	1601.15
28	524	1621.67	70	1764.18	1663	1621.67
32	763	1843.55	85	1914.91	2845	1830.87
34	1258	1936.83	93	2028.06	4170	1961.30
35	1623	2256.53	1953	2406.97	10976	2276.33

**Table 5. Performance evaluation result 2**

to 20. Through the result of this experiment, there is different in the quality of solution found by different methods. Methods such as GA, SA, which do not use local search and need relative smaller runtime, found lower quality solution than other methods. Methods such as ILS, GLS, EDA found high quality solutions but need more time than H-Map.

The advance level experiment does mapping 5 workflows having number of sub-jobs in the range from 25 to 35 sub-jobs. The result of this experiment shows that H-Map algorithm found out higher quality solutions with much shorter runtime than other algorithms in most cases.

## 5 Conclusions

This paper has presented a method, which performs an efficient and precise assignment of heavy communication workflow to Grid resources with respect to SLAs defined deadlines and cost optimization. In our work, the distinguished character is that a sub-job of the workflow can be a sequent or parallel program and a Grid service can handle many sub-jobs at a time. The performance evaluation showed that the proposed algorithm creates solution of equal or better quality than most metaheuristics and needs significantly shorter computation time. The latter is a decisive factor for the applicability of the method in real environments, because large-scale workflows can be planned

and assigned efficiently.

## References

- [1] G. Berriman et al. Montage: a grid enabled image mosaic service for the national virtual observatory. *ADASS*, v. 13, 2003.
- [2] C. Blum et al. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, v. 35 n.3, pages 268–308, 2003.
- [3] I. Brandic et al. Qos support for time-critical grid workflow applications. *Proceedings of e-Science 2005*, 2005.
- [4] L. Burchard et al. The virtual resource manager: An architecture for sla-aware resource management. *Proceedings of the IEEE CCGrid 2004*, IEEE Press, pages 126–133, 2004.
- [5] E. Deelman et al. Mapping scientific workflows onto the grid. *Proc. 2nd European Across Grids Conference*, January 2004.
- [6] M. Hovestadt. Scheduling in hpc resource management systems: queuing vs. planning. *Proc. 9th Workshop on JSSPP at GGF8, LNCS*, pages 1–20, 2003.
- [7] R. Lovas et al. Support for complex grid applications: Integrated and portal solutions. *Proc. 2nd European Across Grids Conference*, May 2004.
- [8] S. Ludtke et al. Eman: Semiautomated software for high-resolution single-particle reconstruction. *Journal of Structure Biology*, v. 128, May 1999.
- [9] S. McGough et al. Making the grid predictable through reservations and performance modelling. *The Computer Journal*, v.48 n.3, pages 358–368, 2005.
- [10] D. Quan et al. Mapping grid job flows to grid resources within sla context. *Proceedings of the European Grid Conference, (EGC 2005)*, 3470:1107–1116, 2005.
- [11] D. Quan et al. On architecture for a sla-aware job flows in grid environments. *Journal of Interconnection Networks, World scientific computing*, pages 245–264, 2005.
- [12] D. Quan et al. Error recovery mechanism for grid-based workflow within sla context. *Accepted by International Journal of High Performance Computing and Networking (IJHPCN)*, 2006.
- [13] L. Richter et al. Workflow support for complex grid applications: Integrated and portal solutions. *Proceedings of the 2nd European Across Grids Conference*, 2004.
- [14] A. Sahai et al. Automated sla monitoring for web services. *DSOM 2002, LNCS 2506*, pages 28–41, 2002.
- [15] D. P. Spooner et al. Local grid scheduling techniques using performance prediction. *IEEE Proc. Computers and Digital Techniques*, pages 87–96, May 2003.
- [16] R. Wolski et al. Experiences with predicting resource performance on-line in computational grid settings. *ACM SIGMETRICS Performance Evaluation Review*, 30(4):41–49, 2003.
- [17] L. Zeng et al. Qos-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, v.30 n.5, pages 311–327, 2004.