Scheduling High-Throughput Computing Applications on the Grid: A Deadline and Budget Constrained Cost-Time Optimisation Algorithm

Rajkumar Buyya[1], Manzur Murshed[2], and David Abramson[3]

[1]Grid Computing and Distributed Systems Laboratory
Dept. of Computer Science and Software Engineering
The University of Melbourne
Parkville, Melbourne, Australia
raj@cs.mu.oz.au

[2] Gippsland School of Computing and Information Technology
Monash University, Gippsland Campus
Churchill, Vic 3842, Australia
Manzur.Murshed@infotech.monash.edu.au

[3] School of Computer Science and Software Engineering
Monash University, Caulfield Campus
Melbourne, Vic 3145, Australia
davida@csse.monash.edu.au

Abstract: Computational Grids and *peer-to-peer* (P2P) networks enable the sharing, selection, and aggregation of geographically distributed resources for solving large-scale problems in science, engineering, and commerce. The management and composition of resources and services for scheduling applications, however, becomes a complex undertaking. We have proposed a computational economy framework for regulating the supply and demand for resources and allocating them for applications based on the users' quality of services requirements. The framework requires economy driven *deadline and budget constrained* (DBC) scheduling algorithms for allocating resources to application jobs in such a way that the users' requirements are met. In this paper, we propose a new scheduling algorithm, called *DBC cost-time optimisation*, which extends the DBC cost-optimisation algorithm to optimise for time, keeping the cost of computation at the minimum. The superiority of this new scheduling algorithm, in achieving lower job completion time, is demonstrated by simulating the World-Wide Grid and scheduling task-farming applications for different deadline and budget scenarios using both this new and the cost optimisation scheduling algorithms.

1 Introduction

Computational Grids [1] and *peer-to-peer* (P2P) computing [2] networks are emerging as next generation parallel and distributed computing platforms for solving large-scale computational and data intensive problems in science, engineering, and commerce. They enable the *sharing*, *selection* and *aggregation* of a wide variety of geographically distributed resources including supercomputers, storage systems, databases, data sources, and specialized devices owned by different organizations. However, resource management and application scheduling is a complex undertaking due to large-scale heterogeneity present in resources, management policies, users, and applications requirements in these environments [17].

A typical world-wide Grid computing environment is shown in Figure 1. In such Grid marketplace and economy, the two key players that come into picture are: *resource owners* (Grid service providers) and *end users* (Grid service consumers). The *resource owners* and *consumers*/*end-users* have different goals, objectives, strategies, and demand patterns. The resources are heterogeneous in terms of their architecture, power, configuration, and availability. They are owned and managed by different organizations with different access policies and cost models that vary with time, users, and priorities. Different applications have different computational models that vary with the nature of the problem. In our earlier work [4]–[7], we investigated the use of economics as a metaphor for management of resources and scheduling applications in Grid computing environments. The computational economy framework provides a mechanism for regulating the supply-and-demand for resources and allocating them to applications based on the users' quality of services requirements [17]. It also offers an incentive to resource owners for sharing resources on the Grid and end-users trade-off between the timeframe for result delivery and computational expenses.

A Grid scheduler, often called *resource broker*, acts as an interface between the user and distributed resources and hides the complexities of Grid computing [4][5]. It performs resource discovery, negotiates for access costs using trading services, maps jobs to resources (*scheduling*), stages the application and data for processing (*deployment*), starts job execution, and finally gathers the results. It is also responsible for monitoring and tracking application execution progress along with adapting to the changes in Grid runtime environment, variation in resource share availability, and failures. Essentially, Grid broker does application scheduling on distributed Grid resources on which it does not have full control—local scheduler has its own policies and performs actual allocation of resource(s) to the user job(s).

.

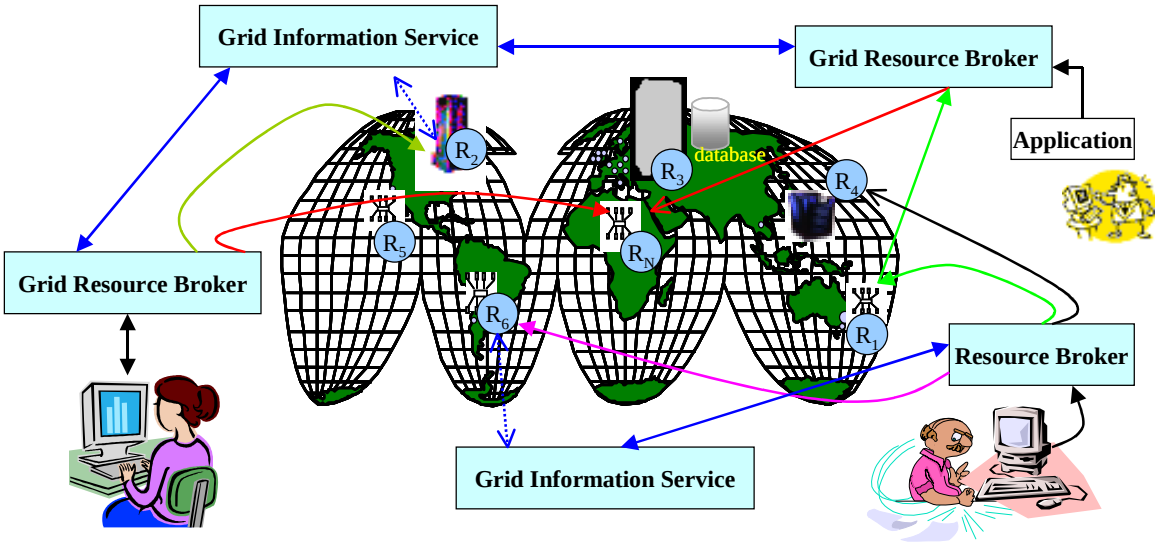


Figure 1: A generic view of World-Wide Grid computing environment.

In our Grid economy framework, the resource brokers use economy driven *deadline-and-budget constrained* (DBC) scheduling algorithms for allocating resources to application jobs in such a way that the users' requirements are met. In our early work [7], we developed three scheduling algorithms for cost, time, and time-variant optimisation strategies that support deadline and budget constraints. We implemented them within the Nimrod-G broker and explored their capability for scheduling task-farming or parameter-sweep and data-intensive computing applications such as drug design [12] on the WWG (World-Wide Grid) [8] testbed resources. To meet users' quality-of-service (QoS) requirements, the broker leases Grid resources and services dynamically at runtime depending on their capability, cost, and availability.

In this work, we propose a new scheduling algorithm, called *DBC cost-time optimisation*, which extends the DBC cost-optimisation algorithm to optimise for time keeping the cost of computation at the minimum. Resources with the same cost are grouped together and time-optimisation scheduling strategy is applied while allocating jobs to a group. We demonstrate the ability of this new scheduling algorithm by implementing it within the economic Grid resource broker simulator built using the GridSim toolkit [3]. The performance of this new algorithm is evaluated by scheduling a synthetic task farming application on simulated WWG testbed resources for different deadline and budget scenarios. We then compare and contrast the results of scheduling with the cost optimisation algorithm.

The rest of this paper is organized as follows. Section 2 discusses application model, scheduling issues, and a new DBC cost-time optimisation scheduling algorithm. The economic Grid broker simulator implemented using the GridSim toolkit and its internal components that simulate and manage the execution of task farming applications are presented in Section 3. The simulation of heterogeneous resources with different capabilities and access costs, creation of synthetic application, and evaluation of proposed cost-time optimisation scheduling algorithm versus the cost optimisation algorithm is discussed in Section 4. The related systems that use different scheduling strategies are being presented in Section 5. The final section presents the summary and conclusion.

2 DBC Cost-Time Optimisation Scheduling

2.1 Application Model and Scheduling

The task-farming and high-throughput computing (HTC) model based parameter-sweep applications, created using a combination of task and data parallel models, contain a large number of independent jobs operating on different data sets. A range of scenarios and parameters to be explored are applied to the program input values to generate different data sets. The programming and execution model of such applications resemble the SPMD (Single Program Multiple Data) model. The execution model essentially involves processing N independent jobs (each with the same task specification, but a different dataset) on M distributed computers where N is, typically, much larger than M. Fortunately, high-throughput parametric computing model is simple, yet powerful enough to formulate distributed execution of many application areas such as: radiation equipment calibration analysis [4], search for extra-territorial intelligence [11], protein folding [21], molecular modelling for drug design [12], human-genome sequence analysis [22], brain activity analysis, high-energy physics events analysis [18], ad-hoc network simulation [19], crash simulation, financial modelling, and Mcell simulations [20]. Therefore, high-throughput *parametric computing* is considered as the *killer application* for the Grid.

Scheduling and orchestrating the execution of parameter sweep applications on world-wide distributed computers appears simple, but complexity arises when users place QoS constraints such as execution completion time (deadline) and computation cost (budget) limitations along with optimisation parameter preference. Such a guarantee of service is hard to provide in a Grid environment since its resources are shared, heterogeneous, distributed in nature, and owned by different organisations having their own policies and charging mechanisms. In addition, scheduling algorithms need to adapt to the changing load and resource availability conditions in the Grid in order to achieve performance and at the same time meet the deadline and budget constraints.

The integration of computational economy as part of a scheduling system greatly influences the way computational resources are selected to meet the user requirements. The users should be able to submit their application along with their requirements to a scheduling system such as Nimrod-G, which can process the application on the Grid on the user's behalf and try to complete the assigned work within a given deadline and cost. The deadline represents a time by which the user requires the result, and is often imposed by external factors like production schedules or research deadlines.

2.2 The Scheduling Algorithm

We have developed a number of algorithms for deadline and budget constrained (DBC) scheduling of task-farming parameter-sweep applications on the Grid. They are: cost-optimisation, time-optimisation, and conservative-time optimisation scheduling algorithms. The performance of these algorithms has been evaluated by implementing them within the Nimrod-G resource broker [3] for scheduling real-world applications and economic-broker simulator [7] through synthetic workloads. A new DBC cost-time optimisation Grid scheduling algorithm is listed in Figure 2.

The DBC cost-time optimisation scheduling algorithm extends the cost-optimisation algorithm to optimise the time without incurring additional processing expenses. This is accomplished by applying the time-optimisation algorithm to schedule task-farming application jobs on distributed resources having the same processing cost. An application containing N jobs along with the deadline and budget constraints are passed as input parameters to the algorithm for processing on M distributed resources/machines. Essentially, the user passes these details to his/her Grid broker that leases resources dynamically at runtime depending on their availability, capability, cost, and user QoS requirements. The performance evaluation of this new algorithm is presented in the Section 4.

Algorithm: DBC_Scheduling_with_Cost_Time_Optimisation(Application: N jobs, Resources: M, Deadline: D, Budget: B)

1. RESOURCE DISCOVERY: Identify characteristics, configuration, capability, and suitability of resources using the Grid information services (GIS).
2. RESOURCE TRADING: Identify *the cost* of all resources and *the capability* to be delivered per cost-unit. The *resource cost* can be expressed in units such as processing cost-per-MI, cost-per-job, CPU cost per time unit, etc. and the scheduler needs to choose suitable unit for comparison.
3. If the user supplies D and B-factors, then determine the absolute deadline and budget based on the capability of resources and their cost, and the application processing requirements (e.g., total MI required).
4. SCHEDULING: Repeat while there exists *unprocessed jobs* and the current time and processing expenses are within the deadline and budget limits. [This step is triggered for each scheduling event or whenever a job completes. The event period is a function of deadline, job processing time, rescheduling overhead, resource share variation, etc.]:

   [SCHEDULE ADVISOR with Policy]

   a. For each resource, predict and establish the *job consumption rate* or *the available resource share* through the measure and extrapolation strategy taking into account the time taken to process previous jobs.
   b. SORT the resources by increasing order of *cost.* If two or more resources have the *same cost*, order them such that powerful ones (e.g., higher job consumption rate or resource share availability, but the first time based on the total theoretical capability, say the total MIPS) are preferred first.
   c. *Create resource groups* containing resources with the same cost.
   d. SORT the *resource groups* with the increasing order of cost.
   e. If any of the resource has jobs assigned to it in the previous scheduling event, but not dispatched to the resource for execution and there is variation in resource availability, then move appropriate number of jobs to the Unassigned-Jobs-List. This helps in updating the whole schedule based on the latest resource availability information.
   f. Repeat the following steps for each resource group as long as there exists unassigned jobs:
      i. Repeat the following steps for each job in the Unassigned-Jobs-List depending on the processing cost and the budget availability: [It uses the *time optimisation strategy*.]
      - Select a job from the Unassigned-Jobs-List.
      - For each resource, calculate/predict the job completion time taking into account previously assigned jobs and the job completion rate and resource share availability.
      - Sort resources by the increasing order of completion time.
      - Assign the job to the first resource and remove it from the Unassigned-Jobs-List if the predicted job completion time is less than the deadline.
5. [DISPATCHER with Policy]

   Repeat the following steps for each resource if it has jobs to be dispatched:

   - Identify the number of jobs that can be submitted without overloading the resource. Our default policy is to dispatch jobs as long as the number of user jobs deployed (active or in queue) is less than the number of PEs in the resource.

Figure 2: Deadline and budget constrained (DBC) scheduling with cost-time optimisation.

3 Implementation of Economic Grid Broker Simulator

The GridSim toolkit is used to simulate a Grid environment and a Nimrod-G like deadline and budget constrained scheduling system called economic Grid resource broker. The simulated Grid environment contains multiple resources and user entities with different requirements. The user and broker entities extend the GridSim class. All the users create experiments containing application specification (a set of

Gridlets that represent application jobs) and quality of service requirements (deadline and budget constraints with optimisation strategy). When the simulation starts, the user entity creates an instance of its own broker entity and passes a request for processing application jobs. We briefly discuss, features of the GridSim toolkit and its usage in the implementation of the economic broker simulator that supports performance evaluation of scheduling algorithms.

3.1 GridSim: A Grid Modeling and Simulation Toolkit

The GridSim toolkit provides a comprehensive facility for simulation of different classes of heterogeneous resources, users, applications, resource brokers, and schedulers [3]. It has facilities for the modeling and simulation of resources and network connectivity with different capabilities, configurations, and domains. It supports primitives for application composition, information services for resource discovery, and interfaces for assigning application tasks to resources and managing their execution. These features can be used to simulate resource brokers or Grid schedulers for evaluating performance of scheduling algorithms or heuristics. We have used GridSim toolkit to create a resource broker that simulates Nimrod-G for design and evaluation of deadline and budget constrained scheduling algorithms with cost and time optimisations.

The GridSim toolkit resource modeling facilities are used to simulate the World-Wide Grid resources managed as time or space-shared scheduling policies. The broker and user entities extend the GridSim class to inherit ability for communication with other entities. In GridSim, application tasks/jobs are modeled as *Gridlet* objects that contain all the information related to the job and its execution management details such as job length in MI (Million Instructions), disk I/O operations, input and output file sizes, and the job originator. The broker uses GridSim's job management protocols and services to map a Gridlet to a resource and manage it throughout its lifecycle. The broker also maintains full details of application scheduling trace data both at coarse and fine levels, which can be used in performance analysis.

3.2 Economic Grid Broker Simulator Architecture

The broker entity architecture and its interaction with other entities is shown in Figure 3. The key components of the broker are: experiment interface, resource discovery and trading, scheduling flow manager backed with scheduling heuristics and algorithms, Gridlets dispatcher, and Gridlets receptor. A detailed discussion on the broker implementation using the GridSim toolkit can be found in [3]. However, to enable the understanding of the broker framework in which the new scheduling algorithm is implemented, we briefly present its operational model:

1. The user entity creates an experiment that contains application description (a list of Gridlets to be processed) and user requirements to the broker via the experiment interface.
2. The broker resource discovery and trading module interacts with the GridSim GIS entity to identify contact information of resources and then interacts with resources to establish their configuration and access cost. It creates a Broker Resource list that acts as a placeholder for maintaining resource properties, a list of Gridlets committed for execution on the resource, and the resource performance data as predicted through the measurement and extrapolation methodology.
3. The scheduling flow manager selects an appropriate scheduling algorithm for mapping Gridlets to resources depending on the user requirements (deadline and budget limits; and optimisation strategy—cost, cost-time, time, and conservative-time). Gridlets that are mapped to a specific resource are added to the Gridlets list in the Broker Resource. In this case, the broker selects the algorithm for DBC cost-time optimisation scheduling.
4. For each of the resources, the dispatcher selects the number of Gridlets that can be staged for execution according to the usage policy to avoid overloading resources with single user jobs.
5. The dispatcher then submits Gridlets to resources using the GridSim's asynchronous service.
6. When the Gridlet processing completes, the resource returns it to the broker's Gridlet receptor module, which then measures and updates the runtime parameter, *resource or MI share available to the user*. It aids in predicting the job consumption rate for making scheduling decisions.
7. The steps, 3–6, continue until all the Gridlets are processed or the broker exceeds the deadline or budget limits. At the end, the broker returns updated experiment data along with processed Gridlets back to the user entity.

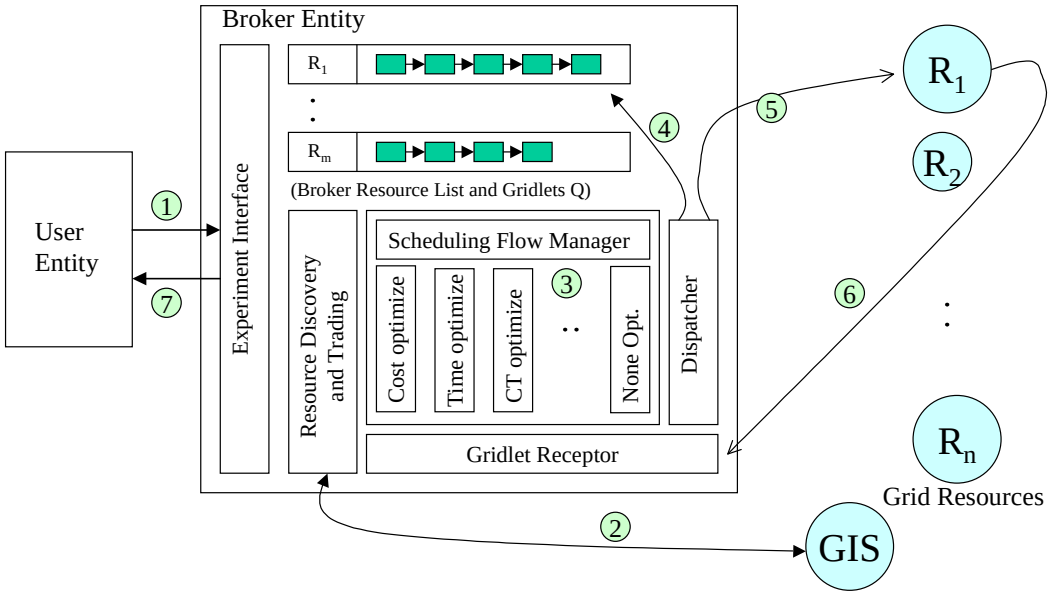


Figure 3: Economic Grid resource broker architecture and its interaction with other entities.

4 Performance Evaluation

To simulate and evaluate application scheduling in GridSim environment using the economic Grid broker requires the modeling and creation of GridSim resources and applications that model jobs as Gridlets. In this section, we present resource and application modeling along with the results of scheduling experiments with quality of services driven application processing.

4.1 Resource Modeling

We modeled and simulated a number of time- and space-shared resources with different characteristics, configuration, and capability as those in the WWG testbed. We have selected the latest CPUs models AlphaServer ES40, Sun Netra 20, Intel VC820 (800EB MHz, Pentium III), and SGI Origin 3200 1X 500MHz R14k released by their manufacturers Compaq, Sun, Intel, and SGI respectively. The processing capability of these PEs in simulation time-unit is modeled after the base value of SPEC CPU (INT) 2000 benchmark ratings published in [9]. To enable the users to model and express their application processing requirements in terms of MI (million instructions) or MIPS (million instructions per second) on the standard machine, we *assume the MIPS rating of PEs is same as the SPEC rating*.

Table 1 shows the characteristics of resources simulated and their PE access cost per time unit in G$ (Grid dollar). The PE capability is derived from their actual SPEC rating and access cost in G$ is artifically defined. The simulated resources resemble the WWG testbed resources used in the Nimrod-G scheduling experiments reported in [10]. The access cost of PE in G$/time-unit not necessarily reflects the cost of processing when PEs have different capability. The brokers need to translate it into the G$ per MI for each resource. Such translation helps in identifying the relative cost of resources for processing Gridlets on them. It can be noted some of the resources in Table 1 have the same MIPS per G$. For example, both R4 and R8 have the same cost and so resources R2, R3, and R10.

4.2 Application Modeling

We have modeled a task farming application that consists of 200 jobs. In GridSim, these jobs are packaged as Gridlets whose contents include the job length in MI, the size of job input and output data in bytes along with various other execution related parameters when they move between the broker and resources. The job length is expressed in terms of the time it takes to run on a standard resource PE with SPEC/MIPS rating of 100. Gridlets processing time is expressed in such a way that they are expected to take at least 100 time- units with a random variation of 0 to 10% on the positive side of the standard resource. That means, Gridlets' job length (processing requirements) can be at least 10,000 MI with a

random variation of 0 to 10% on the positive side. This 0 to 10% random variation in Gridlets' job length is introduced to model heterogeneous tasks similar to those present in the real world parameter sweep applications.

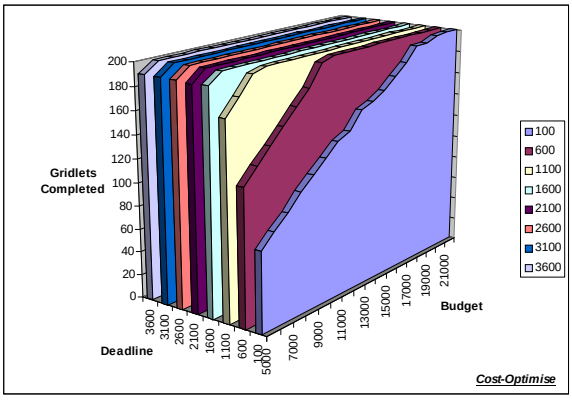
Table 1: World-Wide Grid testbed resources simulated using GridSim.

Resource Name in Simulation	Simulated Resource Characteristics Vendor, Resource Type, Node OS, No of PEs	Equivalent Resource in Worldwide Grid (Hostname, Location)	A PE SPEC/ MIPS Rating	Resource Manager Type	Price (G$/PE time unit)	SPEC/ MIPS per G$
R0	Compaq, AlphaServer, CPU, OSF1, 4	grendel.vpac.org, VPAC, Melb, Australia	515	Time-shared	8	64.37
R1	Sun, Ultra, Solaris, 4	hpc420.hpcc.jp, AIST, Tokyo, Japan	377	Time-shared	4	94.25
R2	Sun, Ultra, Solaris, 4	hpc420-1.hpcc.jp, AIST, Tokyo, Japan	377	Time-shared	3	125.66
R3	Sun, Ultra, Solaris, 2	hpc420-2.hpcc.jp, AIST, Tokyo, Japan	377	Time-shared	3	125.66
R4	Intel, Pentium/VC820, Linux, 2	barbera.cnuce.cnr.it, CNR, Pisa, Italy	380	Time-shared	1	380.0
R5	SGI, Origin 3200, IRIX, 6	onyx1.zib.de, ZIB, Berlin, Germany	410	Time-shared	5	82.0
R6	SGI, Origin 3200, IRIX, 16	Onyx3.zib.de, ZIB, Berlin, Germany	410	Time-shared	5	82.0
R7	SGI, Origin 3200, IRIX, 16	mat.ruk.cuni.cz, Charles U., Prague, Czech Republic	410	Space-shared	4	102.5
R8	Intel, Pentium/VC820, Linux, 2	marge.csm.port.ac.uk, Portsmouth, UK	380	Time-shared	1	380.0
R9	SGI, Origin 3200, IRIX, 4 (accessible)	green.cfs.ac.uk, Manchester, UK	410	Time-shared	6	68.33
R10	Sun, Ultra, Solaris, 8,	pitcairn.mcs.anl.gov, ANL, Chicago, USA	377	Time-shared	3	125.66

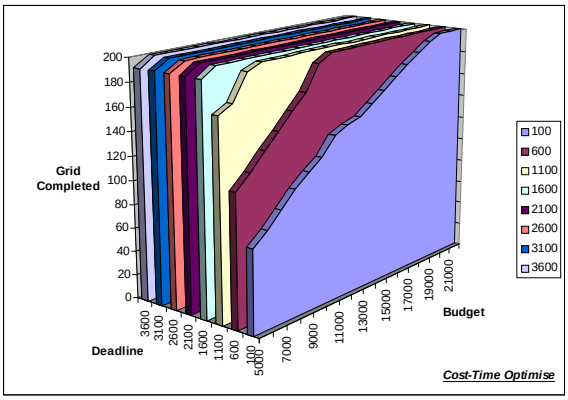
4.3 Scheduling Experiments with Cost and Cost-Time Optimisation Strategies

We performed both cost and cost-time optimisation scheduling experiments with different values of deadline and budget constraints (DBC) for a single user. The deadline is varied in simulation time from 100 to 3600 in steps of 500. The budget is varied from G$ 5000 to 22000 in steps of 1000. The number of Gridlets processed, deadline utilized, and budget spent for the DBC cost-optimisation scheduling strategy is shown in Figure 4a, Figure 4c, and Figure 4e, and for the cost-time optimisation scheduling strategy is shown in Figure 4b, Figure 4d, and Figure 4f. In both cases, when the deadline is low (e.g., 100 time unit), the number of Gridlets processed increases as the budget value increases. When a higher budget is available, the broker leases expensive resources to process more jobs within the deadline. Alternatively, when scheduling with a low budget, the number of Gridlets processed increases as the deadline is relaxed.

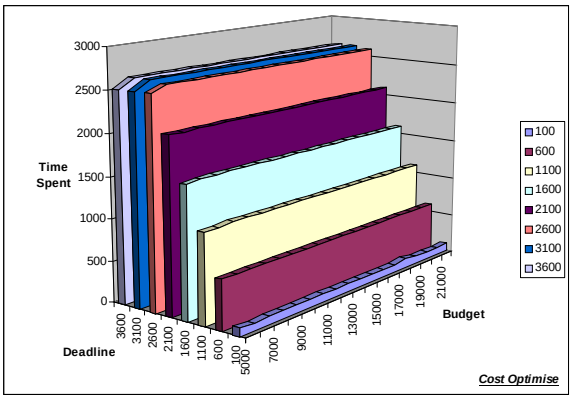
The impact of budget for different values of deadline is shown in Figure 4e and Figure 4f for cost and cost-time strategies. For a larger deadline value (see the time utilization for deadline of 3600), the increase in budget value does not have much impact on resource selection. When the deadline is too tight (e.g., 100), it is likely that the complete budget is spent for processing Gridlets within the deadline.



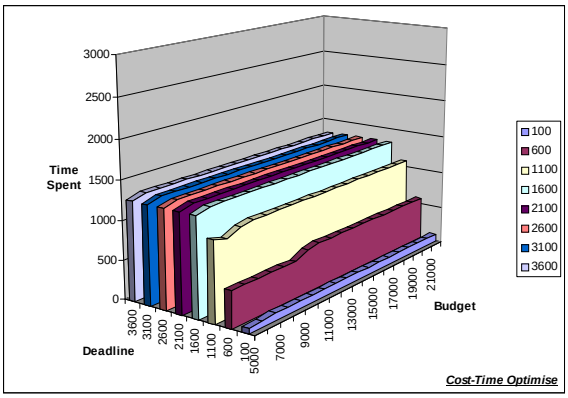
(a) No. of Gridlets processed.



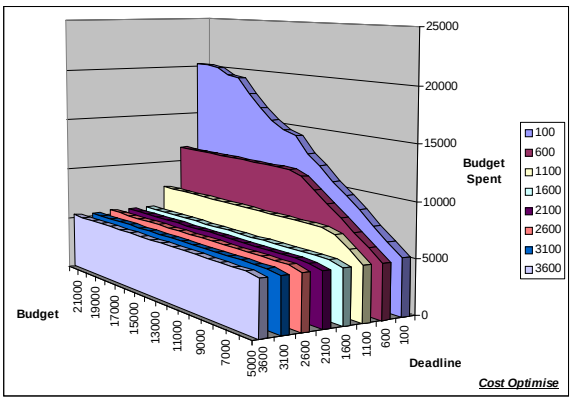
(b) No. of Gridlets processed



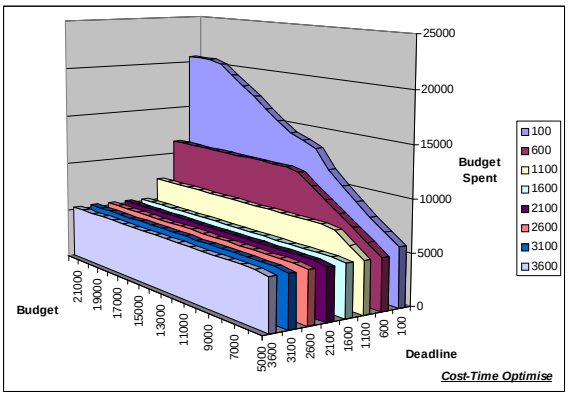
(c) Time spent for processing Gridlets.



(d) Time spent for processing Gridlets.



(e) Budget spent for processing Gridlets.



(f) Budget spent for processing Gridlets.

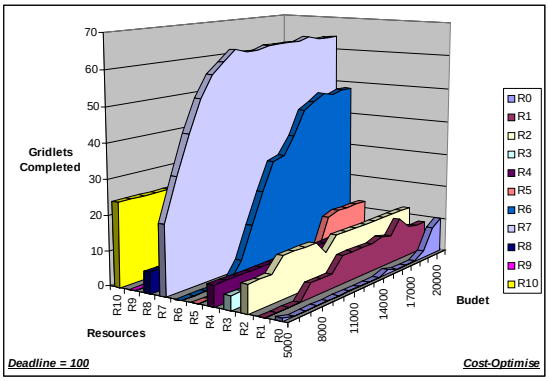
Figure 4: The number of Gridlets processed, time, and budget spent for different deadline and time limits when scheduled using the cost and cost-time optimisation algorithms.

It can be observed that the number of Gridlets processed and the budget-spending pattern is similar for both scheduling strategies. However, the time spent for the completion of all the jobs is significantly different (see Figure 4c and Figure 4d), as the deadline becomes relaxed. For deadline values from 100 to 1100, the completion time for both cases is similar, but as the deadline increases (e.g., 1600 to 3600), the experiment completion time for cost-time scheduling optimisation strategy is much less than the cost optimisation scheduling strategy. This is because when there are many resources with the same MIPS per

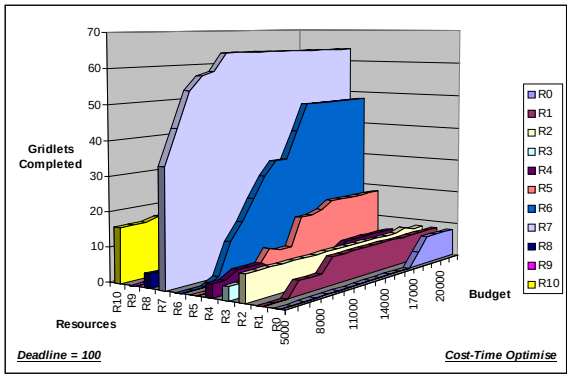
G$, the cost-time optimisation scheduling strategy allocates jobs to them using the time-optimisation strategy for the entire deadline duration since there is no need to spent extra budget for doing so. This does not happen in case of cost-optimisation strategy—it allocates as many jobs that the first cheapest resource can complete by the deadline and then allocates the remaining jobs to the next cheapest resources.

A trace of resource selection and allocation using cost and cost-time optimisation scheduling strategies shown in Figure 5 indicates their impact on the application processing completion time. When the deadline is tight (e.g., 100), there is high demand for all the resources in short time, the impact of cost and cost-time scheduling strategies on the completion time is similar as all the resources are used up as long as budget is available to process all jobs within the deadline (see Figure 5a and Figure 5b). However, when the deadline is relaxed (e.g., 3100), it is likely that all jobs can be completed using the first few cheapest resources. In this experiment there were resources with the same cost and capability (e.g., R4 and R8), the cost optimisation strategy selected resource R4 to process all the jobs (see Figure 5c); whereas the cost-time optimisation strategy selected both R4 and R8 (see Figure 5d) since both resources cost the same price and completed the experiment earlier than the cost-optimisation scheduling (see Figure 4c and Figure 4d). This situation can be observed clearly in scheduling experiments with a large budget for different deadline values (see Figure 6). Note that the left most solid curve marked with the label "All" in the resources axis in Figure 6 represents the aggregation of all resources.

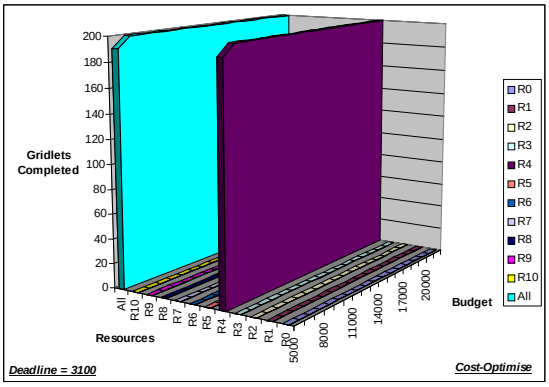
.



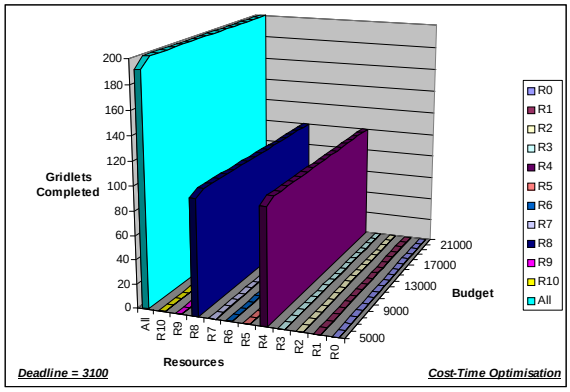
(a) Cost optimisation with a low deadline.



(b) Cost-time optimisation and a low deadline.



(c) Cost optimisation with a high deadline.

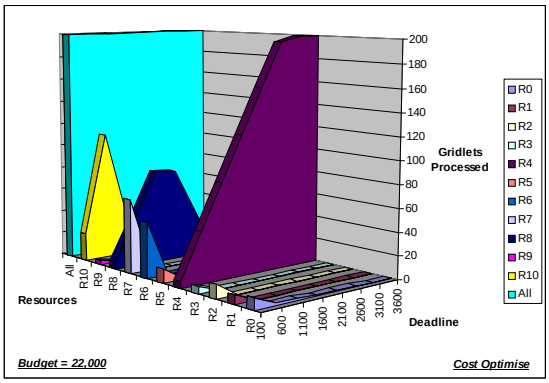


(d) Cost-Time optimisation with a high deadline.

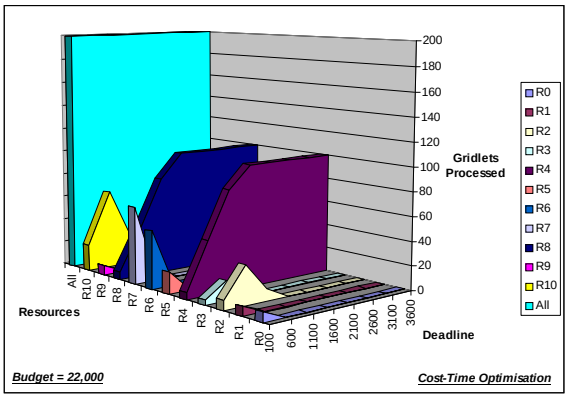
Figure 5: The number of Gridlets processed and resources selected for different budget values with a high deadline value when scheduled using the cost and cost-time optimisation algorithms.

As the deadline increases, the cost optimisation algorithm predominantly scheduled jobs on the resource R4 (see Figure 6a) whereas, the cost-time optimisation algorithm scheduled jobs on resources R4 and R8 (see Figure 6b), the first two cheapest resources with the same cost. Therefore, the application scheduling using the cost-time optimisation algorithm is able to finish earlier compared to the one scheduled using the

cost optimisation algorithm (see Figure 7) and both strategies have spent the same amount of budget for processing its jobs (see Figure 8). The completion time for cost optimisation scheduling continued to increase with increase of the deadline as the broker allocated more jobs to the resource R4 and less to the resource R8. However, the completion time for deadline values 3100 and 3660 is the same as the previous one since the broker allocated jobs to only resource R4. This is not the case with the cost-time optimisation scheduling since jobs are allocated proportionally to both resources R4 and R8 and thus minimizing the completion time without spending any extra budget.



(a) Resource selection when the budget is high.



(b) Resource selection when the budget is high.

Figure 6: The number of Gridlets processed and resources selected for different deadline values with a large budget when scheduled using the cost and cost-time optimisation algorithms.

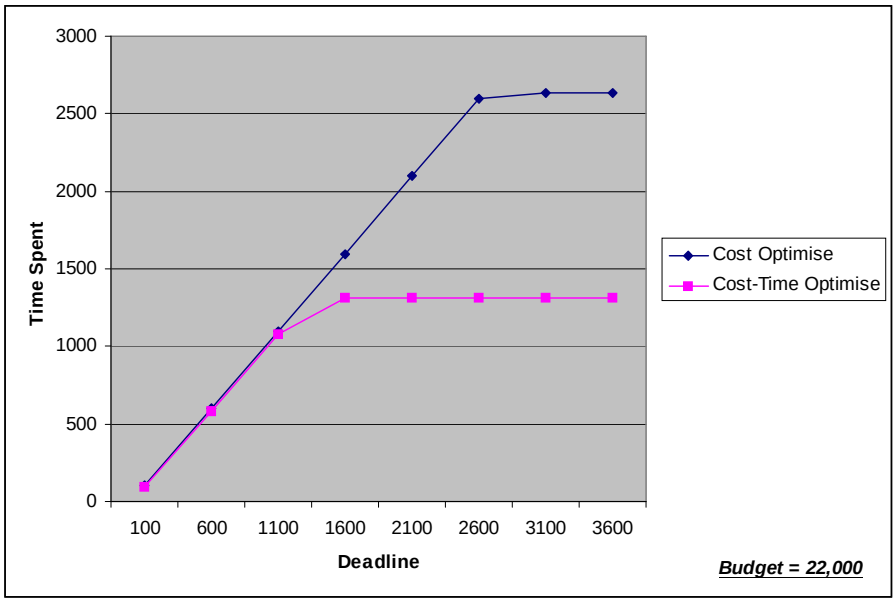


Figure 7: The time spent for processing application jobs for different deadline constraints with a large budget when scheduled using the cost and cost-time optimisation algorithms.

Let us now take a microscopic look at the allocation of resources when a moderate deadline and large budget is assigned. A trace of resource allocation and the number of Gridlets processed at different times when scheduled using the cost and cost-time optimisation algorithms is shown in Figure 9 and Figure 10. It can be observed that for both the strategies, the broker used the first two cheapest resources, R4 and R8 fully. Since the deadline cannot be completed using only these resources, it used the next cheapest

resources R2, R3, and R10 to make sure that deadline can be meet. The cost optimisation strategy allocated Gridlets to resource R10 only, whereas cost-time optimisation allocated Gridlets to resources R2, R3, and R10 as they cost the same price. Based on the availability of resources, the broker predicts the number of Gridlets that each resource can complete by the deadline and allocates to them accordingly (see Figure 11 and Figure 12). At each scheduling event, the broker evaluates the progress and resource availability and if there is any change, it reschedules some Gridlets to other resources to ensure that the deadline can be meet. For example, the broker committed a few extra Gridlets to resource R10 (cost optimisation scheduling strategy, see Figure 11) and to resources R2, R3, and R10 (cost-time optimisation scheduling strategy, see Figure 12) during the first few scheduling events.

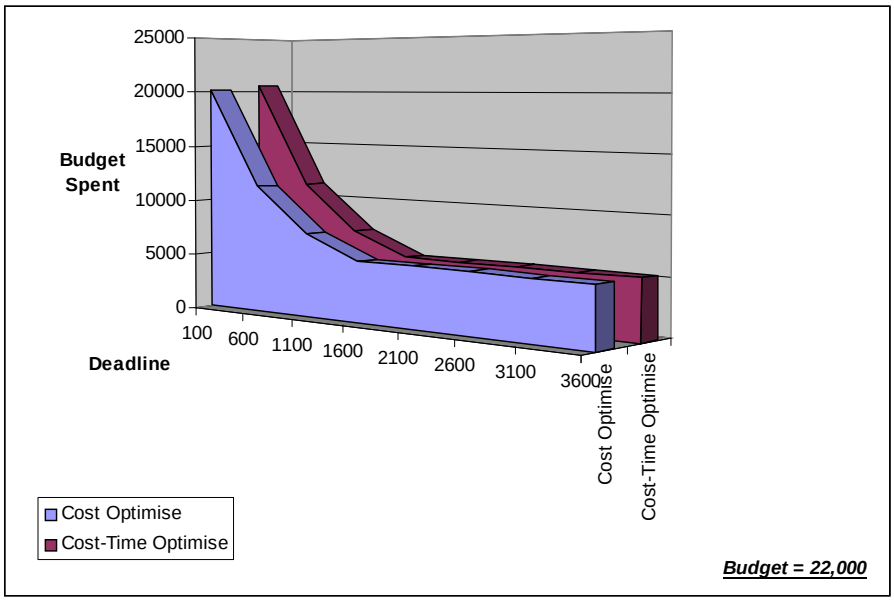


Figure 8: The budget spent for processing application jobs for different deadline constraints with a large budget when scheduled using the cost and cost-time optimisation algorithms.

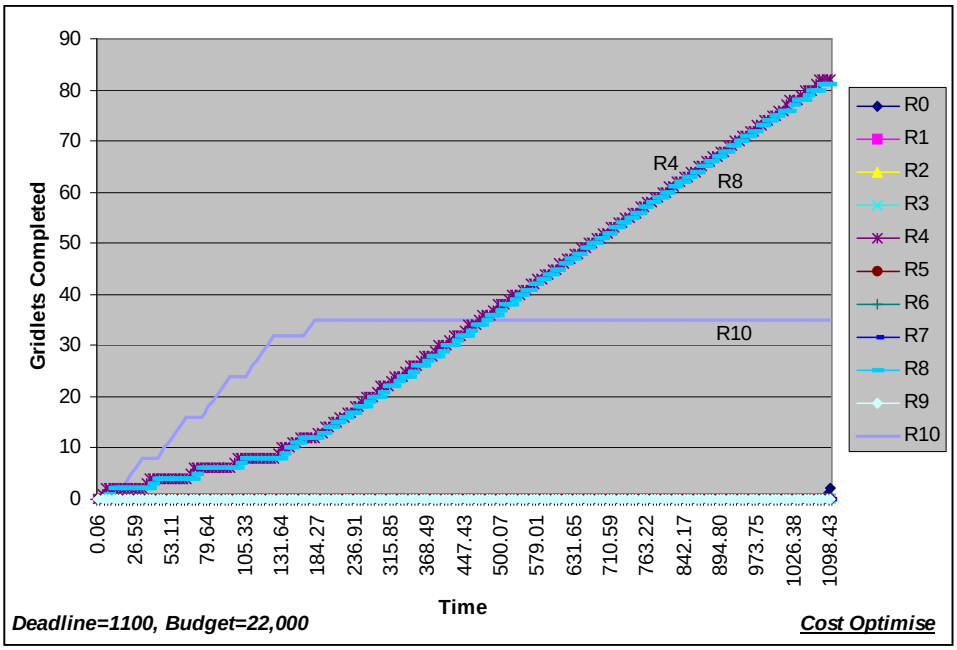


Figure 9: Trace of No. of Gridlets processed on resources for a medium deadline and high budget constraints when scheduled using the cost optimisation strategy.

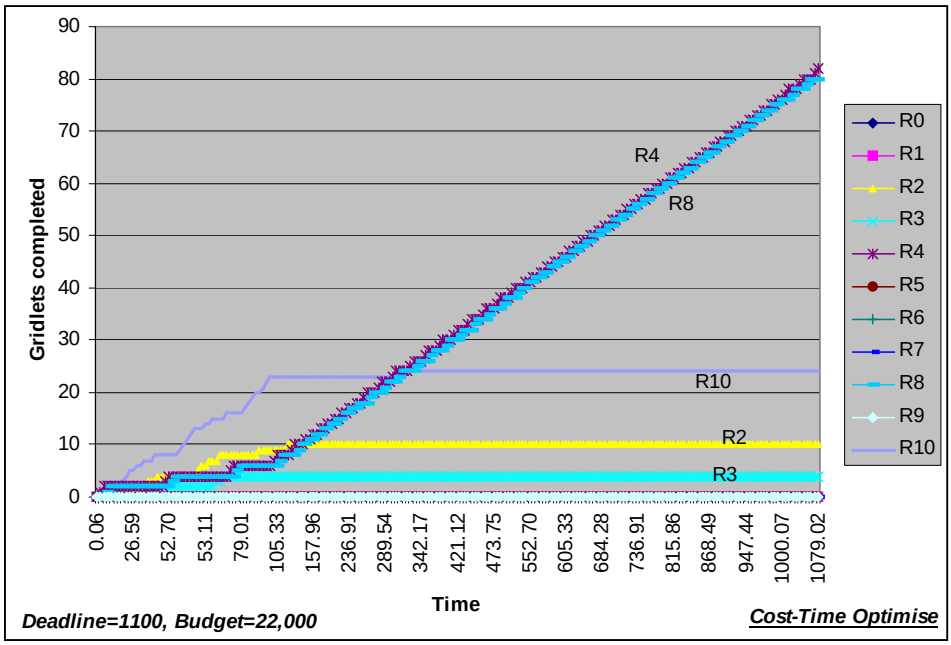


Figure 10: Trace of No. of Gridlets processed on resources for a medium deadline and high budget constraints when scheduling using the cost-time optimisation strategy.

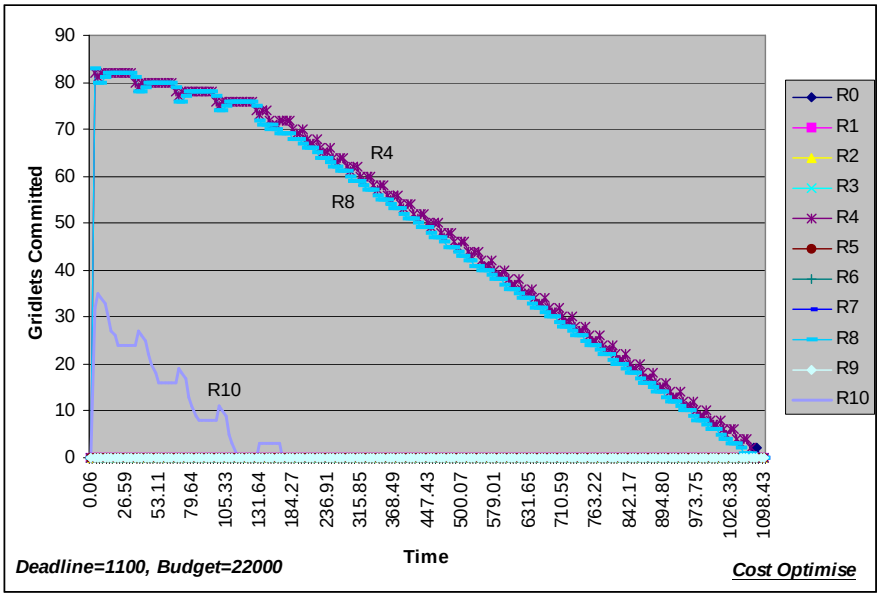


Figure 11: Trace of the number of Gridlets committed to resources for a medium deadline and high budget constraints when scheduled using the cost optimisation strategy.

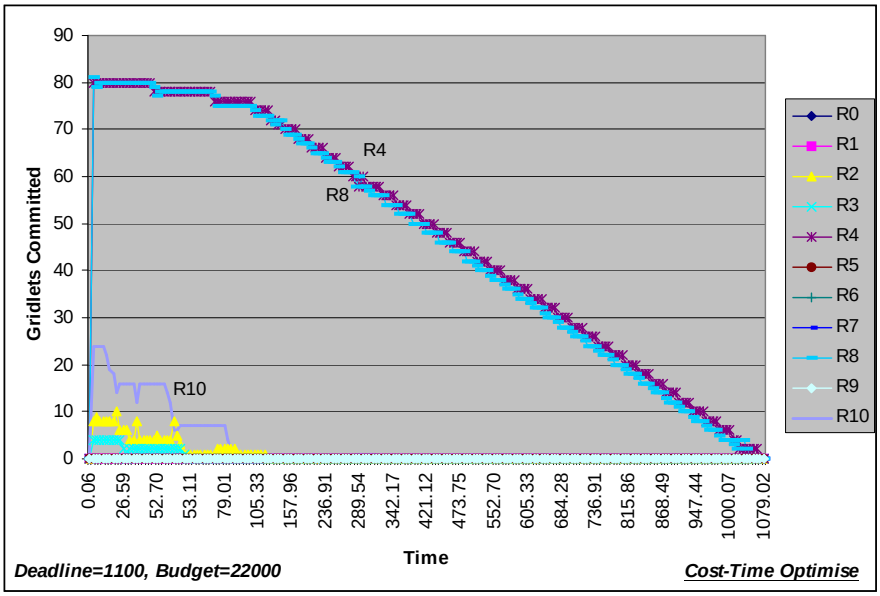


Figure 12: Trace of the number of Gridlets committed to resources for a medium deadline and high budget constraints when scheduled using the cost-time optimisation strategy.

5 Related Work

A number of projects are investigating scheduling on distributed systems [17]. They include Grid scheduling systems such as AppLeS [13], NetSolve [14], and DISCWorld [15], which use system-centric scheduling strategies; and REXEC [16], which supports computational economy-based resource management in a single administrative domain cluster computing environments. The interest in Grid economy is rapidly increasing within the community [24] and projects such as G-Commerce [24] are exploring competitive resource pricing and allocation issues within the Grid environments.

The AppLeS (Application-Level Scheduling) project builds agents for each application responsible for offering a scheduling mechanism [13]. It uses a system-centric scheduling policy, which is targeted for minimising the completion time – it does not take account of the economic cost of jobs processing while selecting resources. A recently developed APST, targeted for parameter-sweep applications, also uses system-centric scheduling strategies [23].

NetSolve is a client-agent-server system, which enables the user to solve complex scientific problems remotely [14]. The NetSolve agent does the scheduling by searching for those resources that offer the best performance in a network. The applications need to be built using one of the APIs provided by NetSolve to perform RPC-like computations. NetSolve also provides an API for creating *task-farming* applications. The scheduling system maps jobs on resources that have appropriate libraries without taking the cost/price for processing jobs on them into consideration.

DISCWorld (Distributed Information Systems Control World) is a service-oriented metacomputing environment, based on the client-server-server model [15]. Remote users can login to this environment over the Internet and request access to data, and also invoke services or operations on the available data. DISCWorld aims for remote information access. The scheduling strategies used in the DISCWorld system are also system-centric in nature.

Another related tool that uses the concept of computational economy is REXEC, a remote execution environment [16] for co-operative distributed systems such as clusters with a centralized scheduling manager. At the command line, the user can specify the maximum rate (credits per minute) that he is willing to pay for CPU time. The REXEC client selects a node that fits the user requirements and executes the application on it. The REXEC system provides an extended shell for remote execution of applications on clusters. Its scheduling strategies are targeted for centralised systems and the allocation of resource share is proportional to the users valuation of their jobs. Whereas, in our Grid resource broker, scheduling strategies are targeted for geographically distributed systems—each resource has its own scheduler that

performs actual allocation of resources to user jobs. That means, cluster schedulers use cooperative computing economy since they aim for global optimisation, whereas Grid schedulers use competitive computing economy since every entity aims to optimise its own objectives.

6 Summary and Conclusion

Computational Grids enable the sharing, discovery, selection, and aggregation of geographically distributed heterogeneous resources for solving large-scale applications. We proposed computational economy as a metaphor for managing the complexity that is present in the management of distributed resources and allocation. It allows allocation of resources depending on the users' quality of service requirements such as the deadline, budget, and optimisation strategy. This paper proposed a new deadline and budget constrained scheduling algorithm called *cost-time optimisation*. We developed a scheduling simulator using the GridSim toolkit and evaluated the new scheduling algorithm by comparing its performance and quality of service delivery with the cost optimisation algorithm.

When there are multiple resources with the same cost and capability, the cost-time optimisation algorithm schedules jobs on them using the time-optimisation strategy for the deadline period. From the results of scheduling experiments for many scenarios with a different combination of the deadline and budget constraints, we observe that applications scheduled using the *cost-time* optimisation are able to complete earlier than those scheduled using the cost optimisation algorithm without incurring any extra expenses. This establishes the superiority of the new deadline and budget constrained cost-time optimisation algorithm in scheduling jobs on global Grids.

Efforts are currently underway to implement the cost-time optimisation algorithm within the Nimrod-G Grid resource broker for scheduling parameter sweep applications on the World-Wide Grid testbed resources.

Software Availability

The GridSim toolkit and the economic Grid broker simulator with source code can be downloaded from the GridSim project website:

```
http://www.buyya.com/gridsim/
```


Acknowledgements

We thank Rob Gray (DSTC) for his comments on improving the paper and Marcelo Pasin (Federal University of Santa Maria, Brazil) for his help in modeling resources with SPEC benchmark ratings.

References

[1] I. Foster and C. Kesselman (editors), *The Grid: Blueprint for a Future Computing Infrastructure*, Morgan Kaufmann Publishers, USA, 1999.

[2] A. Oram (editor), *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly Press, USA, 2001.

[3] R. Buyya and M. Murshed, *GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing*, The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, 2002 (to appear).

[4] D. Abramson, J. Giddy, and L. Kotler, *High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?*, Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2000), May 1-5, 2000, Cancun, Mexico, IEEE CS Press, USA, 2000.

[5] R. Buyya, D. Abramson, and J. Giddy, *Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid*, Proceedings of the 4th International Conference and Exhibition on High Performance Computing in Asia-Pacific Region (HPC ASIA 2000), May 14-17, 2000, Beijing, China, IEEE CS Press, USA, 2000.

[6] R. Buyya, D. Abramson, and J. Giddy, *An Economy Driven Resource Management Architecture for Global Computational Power Grids*, Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), June 26-29, 2000, Las Vegas, USA, CSREA Press, USA, 2000.

[7] R. Buyya, J. Giddy, and D. Abramson, *An Evaluation of Economy-based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications*, Proceedings of the 2nd International Workshop on Active Middleware Services (AMS 2000), August 1, 2000, Pittsburgh, USA, Kluwer Academic Press, USA, 2000.

[8] R. Buyya, *The World-Wide Grid*, http://www.buyya.com/ecogrid/wwg/

[9] SPEC, *SPEC CPU2000 Results*, http://www.specbench.org/osg/cpu2000/results/cpu2000.html, accessed on Jan. 30, 2002.

[10] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, *Economic Models for Resource Management and Scheduling in Grid Computing*, The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, 2002 (to appear).

[11] W. T. Sullivan, III, D. Werthimer, S. Bowyer, J. Cobb, D. Gedye, and D. Anderson, *A new major SETI project based on Project Serendip data and 100,000 personal computers*, Proceedings of the Fifth International Conference on Bioastronomy, 1997. http://setiathome.ssl.berkeley.edu/

[12] R. Buyya, K. Branson, J. Giddy, and D. Abramson, *The Virtual Laboratory: Enabling Molecular Modelling for Drug Design on the World Wide Grid*, Technical Report, CSSE-103, Monash University, December 2001.

[13] F. Berman, and R. Wolski, *The AppLeS Project: A Status Report*, Proceedings of the 8th NEC Research Symposium, Germany, May 1997.

[14] H. Casanova, M. Kim, J. Plank, and J., Dongarra, *Adaptive Scheduling for Task Farming with Grid Middleware*, The International Journal of High Performance Computing, Vol. 13, No. 3, Fall, 1999.

[15] K. Hawick, H. James, et. al., *DISCWorld: An Environment for Service-Based Metacomputing*, Future Generation Computing Systems (FGCS), Vol. 15, 1999.

[16] B. Chun and D. Culler, *User-centric Performance Analysis of Market-based Cluster Batch Schedulers*, 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2002), Berlin, Germany, May 2002.

[17] R. Buyya, *Economic-based Distributed Resource Management and Scheduling for Grid Computing*, PhD Thesis, Monash University, Melbourne, Australia, April 12, 2002 (submitted). Online at: http://www.buyya.com/thesis/thesis.pdf

[18] R. Buyya et. al., *The HEPGrid (High Energy Physics and the Grid Network) Project*, http://www.buyya.com/hepgrid, 2001.

[19] D. Abramson, K. Power, and R. Sosic, *Simulating Computer Networks using Clusters of PCs*, HPC-TelePar'99 at the 1999 Advanced Simulation Technologies Conference (ASTC'99), April 11-15, San Diego, California.

[20] H. Casanova, T. Bartol, J. Stiles, and F. Berman, *Distributing MCell Simulations on the Grid*, The International Journal of High Performance Computing and Supercomputing Applications, Volume 15, Number 3, Fall 2001.

[21] V. Pande, *Protein Folding on Internet-wide distributed computing (Folding@Home project*), http://www.stanford.edu/group/pandegroup/Cosm/, Stanford University, 2001

[22] Blackstone, *Post-Genomic Challenges Drive Life Sciences to High Throughput Computing (HTC)*, White Paper, April 2001, http://www.blackstonecomputing.com/products/whitePapers/

[23] H. Casanova, G. Obertelli, F. Berman, and R. Wolski, *The AppLeS Parameter Sweep Template: User-Level Middleware for the Grid*, Proceedings of the IEEE SC 2000: International Conference Networking and Computing, Nov. 2000, Dallas, Texas.

[24] T. Hacker and W. Thigpen, *Accounting Models Research Group*, Global Grid Forum, http://www.gridforum.org/5_ARCH/ACCT.htm

[25] R. Wolski, J. Plank, J. Brevik, and T. Bryan, *Analyzing Market-based Resource Allocation Strategies for the Computational Grid*, International Journal of High-performance Computing Applications, Volume 15, Number 3, Sage Publications, USA, Fall 2001.