

Fuzzy Grid Scheduling Using Tabu Search

Carole Fayad, Jonathan M. Garibaldi and Djamila Ouelhadj

Abstract— This paper considers the problem of grid scheduling in which different jobs are assigned to different processors, and a scheduling algorithm is devised, using tabu search, to find optimal solutions in order to maximize the number of scheduled jobs. However, inherent in the nature of the application, the processing times of jobs are not precise but are estimates that vary between minimal values, in case of premature failure of jobs, to maximal values as specified ‘a priori’ by well-experienced users. Fuzzy methodology becomes instrumental in this application as it allows the use of fuzzy sets to represent the processing times of jobs, modelling their uncertainty. This work presents the implementation of a tabu search algorithm to create good schedules and explores the robustness of the schedule when processing times do vary by assessing its performance in both fuzzy and crisp modes. Finally, the impact of changing the shapes of fuzzy completion times and the average job length on the schedule performance is discussed.

I. INTRODUCTION

Scheduling jobs in a Grid computing environment is a complex problem as resources are geographically distributed based on their different usage policies as well as different loads and availability [1]. In Grid computing architectures, there are two types of schedulers, Super Schedulers (SSs) and Local Schedulers (LSs). Local Schedulers, also referred to as clusters, form part of the Super Schedulers. Normally, jobs are submitted to the SSs that dispatch them to the LSs, which are then responsible for allocating resources in the cluster to the incoming jobs. Therefore, the use of a distributed scheduling system and negotiation models should prove more efficient than that of a monopoly attempting to capture the entire Grid’s information at a single point. Service Level Agreements (SLAs) are one major approach to solve resource provisioning in Grid scheduling, establishing an end-to-end negotiation between the user and the scheduler which contains information on the jobs to be executed [2]. Scheduling at the level of SSs and LSs are well-established areas of research, but this paper focuses on local scheduling.

At the LSs level, it is common practice for Grid scheduling systems to adopt batch-based heuristics for scheduling. However, these systems fail to find the best scheduling solution, for the reason that they provide only one level of service, namely ‘run this when it gets to the head of the queue’. This strategy, although it might provide occasional ‘good’ solutions, can fall short of identifying the best solutions, which can often be obtained by using high-level meta-heuristic methods. Tabu search [3], simulated annealing [4] and genetic algorithms [5]) are examples of meta-heuristic approaches that can improve the local search algorithms to

escape local optima by either accepting worse solutions, or by generating good starting solutions for the local search in a more intelligent way than just providing random initial solutions. Comparing batch-based heuristics with meta-heuristics, Jarvis et al. have aimed to minimise a schedule’s makespan, and their study found that batch queuing under-achieved when compared with genetic algorithms [6]. Braun et al. compared the performance of batch queuing heuristics, tabu search, genetic algorithms, and simulated annealing in their ability to minimise the makespan [7]. Their results showed that the batch queuing heuristics achieved the worst results. Nevertheless, despite the better performance of such meta-heuristic scheduling methods, there is still little use of the latter methods to generate good quality schedules [8] in Grid scheduling.

In Grid scheduling, jobs consist of multiple computational tasks to be assigned to distributed resources, while still respecting any dependence in the workflow, therefore requiring accurate information and precise job processing, start and end times. This is because, in real-world applications, after having been started on one or more processors, jobs might fail immediately or after an elapsed time. Moreover, it is not always feasible to measure the exact time needed to finish one job as it can vary from one instantiation to another.

In this paper, a novel heuristic scheduling approach is investigated with the capability of dynamically and efficiently matching flexible user requirements with available resources, featuring uncertain processing times. Fuzzy techniques have features that make them an attractive tool to address this problem, as they allow the representation of uncertain data and provide a rich set of tools to permit their manipulation in order to generate fuzzy schedules. In the fuzzification process, crisp data (supplied by the user in their SLA request) was transformed into a fuzzy representation using fuzzy sets to represent the actual uncertainties of the problem. A tabu search meta-heuristic was then applied in order to search for an efficient scheduling of the SLAs at the local level of the Grid resources. Such a schedule would be most likely to be used to decide whether to accept the SLA request and hence to form a contract with the user to run the job, and not to determine the order in which the SLAs are actually executed on the Grid resource.

Section II introduces the fuzzy techniques used in this application. Section III give a general background of the tabu search meta-heuristic method, while Section IV explains how tabu search was implemented in this application of fuzzy grid scheduling. Section V details the problem representation adopted the modelling process. Section VI present the experimental methods used and results obtained. Finally, Section VII discusses the implications of the findings.

The authors are with the School of Computer Science and IT, University of Nottingham, Wollaton Road, Nottingham, NG8 1BB, UK (phone: +44 115 9514216; email: jmg@cs.nott.ac.uk).

II. FUZZY METHODOLOGY

A. Fuzzy Sets

Unlike a conventional crisp set, which enforces either membership or non-membership of an object in a set, a fuzzy set allows grades of membership in the set. A fuzzy set, A , is defined by a membership function, $\mu_A(x)$, which assigns to each object x in the universe of discourse X , a value representing its grade of membership in this fuzzy set: $\mu_A(x) \rightarrow [0, 1]$. Different shapes of membership function can be used; conventionally, the choice of the shape is subjective [9], [10].

B. Fuzzy Processing Times, Start Times and Completion Times

Processing times of jobs are uncertain, due to the nature of the domain where some jobs might take longer to process in some instances, and some complete in shorter times to that expected in other instances. Furthermore, a job execution might fail altogether; often a failure will occur almost immediately (due to some incompatibility between the job and the host machine).

Moreover, with uncertain processing times, the completion time of jobs is consequently uncertain and the starting time for the consecutive jobs on the same processor becomes uncertain too.

In general, for a job J_i , uncertain processing times \tilde{p}_i are generally modelled by triangular fuzzy set (p_i^1, p_i^2, p_i^3) , where it is increasingly likely that the processing time falls in the range $[p_i^1, p_i^2]$ and decreasingly likely it falls in the range $[p_i^2, p_i^3]$. The same applies when triangular-shaped fuzzy sets are used to represent uncertain completion times, the reason being that dealing with fuzzy processing times will result in fuzzy completion times (Fig. 1).

C. Fuzzy Operators

The two fuzzy operations, addition and subtraction of two fuzzy numbers $A = (a1, a2, a3)$ and $B = (b1, b2, b3)$, used in this work are defined as:

$$A + B = (a1 + b1, a2 + b2, a3 + b3),$$

$$A - B = (a1 - b3, a2 - b2, a3 - b1).$$

III. TABU SEARCH

Glover first proposed ‘tabu search’ in 1977 as an optimization methodology applicable to non-linear problems. It operates as an iterative improvement procedure that starts from some initial feasible solution and attempts to determine a better solution in the manner of the greatest-descent algorithm [11].

The problem is hence converted into a combinatorial optimisation problem defined by a pair (F, E) , where F is the finite set of the admissible points (configurations) and E is the cost function to be minimised, also called the *error* or *energy* function:

$$E : F \rightarrow \mathfrak{R}.$$

The best solution is then searched using a heuristic procedure, the latter biased toward points of lower energy. The possible transition from one configuration to another is referred to as ‘set of moves’.

The system is designed in a way as such to discourage the occurrence of ‘limit cycles’ and the confinement of the trajectory in a limited portion of the search. A modified greedy approach is used for this purpose, where different solutions are assessed, with different elementary moves and the ones that give the lowest value of E are selected. Also, in order to avoid obtaining limit cycles, the algorithm prohibits the execution of moves that can reverse (or undo) the effect of moves produced recently. Therefore, those reversing moves become ‘tabu’. By making use of the short-term memory of the recent solutions in a tabu list, the optimization routines can escape from local optima. A diversification strategy is usually also added to avoid confinement of the search trajectory [12].

IV. TABU SEARCH IMPLEMENTATION OF FUZZY GRID SCHEDULING

A Grid resource consists of a set of m processors or hosts (‘machines’ in standard scheduling terms). In local Grid scheduling, a set of n SLAs (jobs), J_i ($i = 1, \dots, n$), have to be scheduled on m suitable machines, M_j ($j = 1, \dots, m$). For each job, J_i , we assume an estimation of the processing time taken by M_j to execute J_i . This processing time is fuzzy and is denoted as \tilde{p}_i . Each job has an earliest possible start time, termed the *release time*, r_i of J_i and its latest finish time, termed the *due time*, d_i . The fuzzy completion time of job J_i is \tilde{C}_i . The fuzzy tardiness of J_i is defined by $\tilde{T}_i = \max(0, \tilde{C}_i - d_i)$.

The objective in this study was to minimise the number of tardy jobs, usually termed the *unit penalty*, U_i , where

$$U_i = \begin{cases} 1 & T_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

Formally, this scheduling problem is denoted as $R \mid r_i \mid \sum U_i$; it is known to be *NP-hard* [13]. In practice, each late job would have an associated penalty cost specified in its SLA; in effect, a cost for breaking the service level agreement. Thus, the problem has *weighted* unit penalty as the objective function, and would be formally denoted as $R \mid r_i \mid \sum w_i U_i$.

To solve this combinatorial optimisation-scheduling problem, a tabu search method with short-term memory was developed and used herein. The initial solution was generated using the well-known batch queuing heuristics FCFS, Min-min, Max-min, sufferage, and backfilling. Then the solution was improved by using two moves: SLA-swap and SLA-transfer moves. The SLA-swap move swaps two SLAs performed by different processors. SLA-transfer move shifts the SLA to another processor. The search evaluated all the possible moves, and then selected the best move to apply, even if the move yielded a worse objective value than the current one until the stopping criterion is reached. Tabu search makes use of a tabu list, where it stores all applied

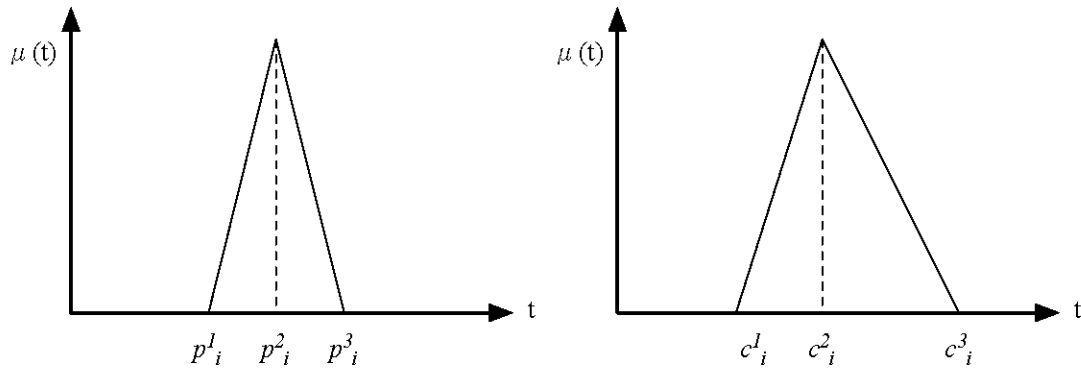


Fig. 1. Fuzzy processing times and fuzzy completion times

moves in order to avoid cycles, by forbidding a move which reverses a recent one. In practice, each tabu move was initially given a maximum count number, which was the chosen number of forbidden iterations. Then, each time the list was updated, the count number of each tabu move was decreased by 1, and on reaching zero, the move was removed from the tabu list, thus becoming valid. When the best move was selected, the tabu list was checked to find out if the move existed. If the best move was not tabu, it was applied to the current solution and then added to the list. If the best move was tabu, then it could not be applied unless the aspiration criterion held, whereby the resulting objective value was higher than that of the best solution so far seen. If the criterion failed, then the best move was discarded, the next best move was considered, and the process was repeated [14], [12].

V. PROBLEM REPRESENTATION

A. Data Inputs

The data used in this case-study had been collected by computer scientists working on Grid Systems, at the University of Manchester. A number of parameters had been identified to be of interest to the conducted research and had therefore been provided as input data. Those parameters associated with each job are described below, as follows:

- JobID: A unique ID is associated with each job.
- Execution time (also referred to as processing time): This is the time required to finish processing of the job. It is uncertain and provided as an upper limit value.
- Release time: This is the date/time the job has been released to the Grid and hence this is the earliest possible time the job is allowed to commence execution.
- Due time: This is also referred to as the latest finishing time; it is the latest possible time before which the job should finish execution.
- CPU#: This value denotes the number of CPUs required by the job and therefore CPUs allocated to a job must be solely preserved for that job during its execution. It is assumed that the overall number of CPUs available

is 64 processors; although the largest number of CPUs needed at any time by one job is 40.

(Execution time, release time and due time are given in time units.)

Note that the capacity constraint applied herein, whereby one processor can process only one job at one time. There was also a job-related constraint, where all operations of one job must be executed on the processors required, and therefore specified by the scheduler, at the same starting date.

Moreover, inherent of the nature of this Grid scheduling problem is that tardy jobs are not to be scheduled and are therefore removed off the processors. In other words, jobs are submitted to the scheduler, a tabu search is then applied to find best solutions. Jobs, found to be tardy, are not scheduled on the Grid. This is due to the fact that a job that finishes past its due date is deemed of no use to the user.

Another constraint to abide by in this case-study was that time limits were defined by the Grid scientists to lie between a minimum of 1 time unit and a 148 time units. This is considered as a virtual wall-clock time. Therefore, all jobs are to be scheduled within this 64x148 rectangular area and a job is confirmed to be scheduled only when it is finishing or when its completion time falls short of the 148 time units. Moreover, the use of an upper bound to the fuzzy job execution time ensures that jobs do not exceed the upper limit of 148 time units, at any time.

B. Fuzzy Criteria

The developed scheduling algorithm aims to maximise the number of scheduled jobs, where a job is scheduled if it completes prior to a given due date, i.e. if job is not tardy. But in scheduling applications, it might be the case that not all jobs complete in time and some are therefore tardy. Tardiness is normally evaluated by comparing a job's completion time with its due date. In crisp applications, a straightforward subtraction applies:

$$T_i = \max(0, C_i - d_i).$$

However, in fuzzy applications, where simple subtraction cannot apply, methods to compare fuzzy sets ought to be used

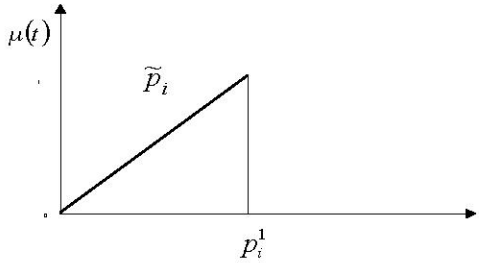
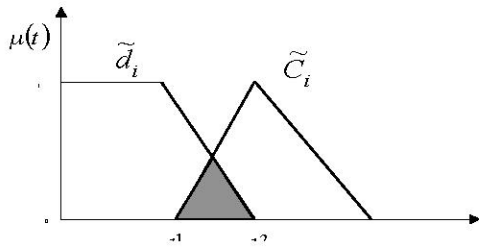


Fig. 3. Proposed representation of fuzzy processing time with and zero lower limit

instead. The ‘Area of Intersection’ introduced in this by Sakawa [15] is used herein to measure the portion that is completed by the due date \tilde{d}_i (Fig. 2), hence an indication of how likely a job is to be tardy. The ratio of tardiness, RT_i of job J_i is defined as:

$$RT_i = \frac{\text{area}(\tilde{C}_i \cap \tilde{d}_i)}{\text{area}(\tilde{C}_i)}$$

Furthermore, we make the assumption that job J_i is scheduled (i.e. not tardy) if $RT_i > 0.5$ and, conversely, job J_i is not scheduled (i.e. tardy), if $RT_i < 0.5$.

C. New Definition of Tardiness

In contrast to the situation described above, in this case study the values for processing times are provided as upper limit values. On the other hand, a job J_i might fail to execute and drop out at a premature stage. This can happen at any time of job execution, although it seems it is the case that the likelihood of the job failing decreases the longer the job is in process. A job, failing on one processor, is subsequently cancelled on all other processors the job has been allocated to. Therefore, to represent the fuzzy processing time in such a way as to allow for an upper limit and to allow for risk of job failure at its early start time, the fuzzy representation shown in Fig. 3 was proposed.

As for the job’s due time, it is given a crisp representation, referred to as the latest finish time. In this application, rectangular (trapezoidal) fuzzy sets are used for modelling of the due time, as an implication that jobs are expected to complete processing at a *specific* time prior to the given deadline. The fuzzy set representation for due time is shown in Fig. 4.

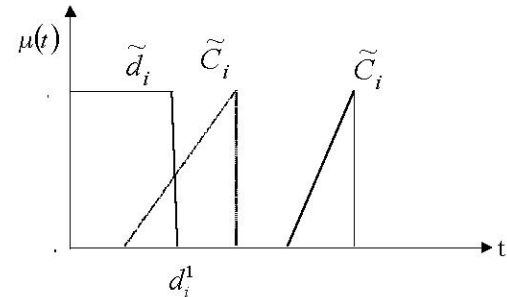
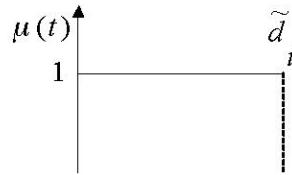


Fig. 5. Sakawa’s ratio of tardiness

For each job J_i , the Sakawa concept applies accordingly, where the ratio of tardiness measures the intersection between the two fuzzy sets of due date and completion (see Fig. 5). The interesting questions to examine are (i) how the schedule performs having fuzzified the problem (completion times + criterion)? and (ii) how do the changes of the shape and of the size of the production times affect the solution? These issues are addressed below.

VI. EXPERIMENTATION

A. Methods

The purpose of these experiments was to validate how the concept of representing uncertain processing times using fuzzy sets could be used in an attempt to demonstrate the benefits this may have on the performance of the schedule. The parameters used for the tabu search are listed in Table I.

Three sets of data were used for validation (Table II). Those sets contain different cases of bulk job submission, obtained from the Grid centre at the University of Manchester. Four methods of validation were examined:

TABLE I
TABU SEARCH PARAMETERS

Maximum number of iterations	200
Maximum number of extra iterations	15
Size of tabu list	10
Tabu search moves	transfers swaps shifts
Is intensification in use	yes
Criterion to be maximised	number of scheduled jobs

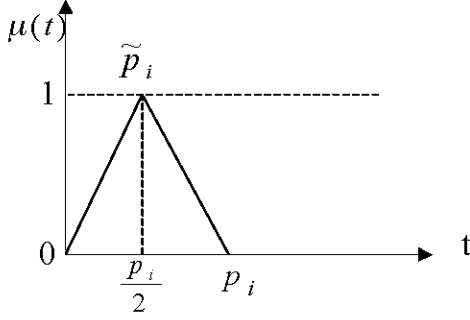


Fig. 6. Modified shape of the fuzzy set

TABLE II
SUMMARY OF THE DATA SETS

	Processing Time	Number of Processors (Hosts)	Earliest Release Time	Latest Finish Time
Case-study A				
Min	1	1	1	15
Max	40	40	130	148
Average	6	5	43.7	95.8
Case-study B				
Min	1	1	1	15
Max	40	40	130	148
Average	6	5	43.7	95.8
Case-study C				
Min	1	1	1	15
Max	40	40	130	148
Average	6	5	43.7	95.8

- 1) Running the algorithm with crisp values.
- 2) Running the algorithm with the fuzzy processing times as presented in Section V-C. The processing times of jobs were modelled using triangular shaped fuzzy sets (Fig. 3), where the centre value and upper bound are equivalent to the provided input value of processing time. The lower bound was set to zero, to account for the possibility of the job's failure to execute.
- 3) Running the algorithm with modified shape of the fuzzy processing time. This was done in the aim to investigate the performance of the schedule with different shapes of fuzzy sets. Here, the lower bound was set to zero and the upper bound took the given value, but the centre of the set was equal to half value of the given upper limit, as shown in Fig. 6.
- 4) Running the algorithm by repeating all above (steps a, b and c) with enlarged fuzzy sets. The sets were enlarged by multiplying the values of the processing times by an arbitrary value of 5. The reason for this was that we noticed that the values given for processing times were much smaller in comparison to the due times and we wanted to test the robustness of the algorithm if complexity of the problem was to increase by increasing the values of processing times.

TABLE III
RESULTS FOR EACH CASE STUDY

Case Study (Number of Jobs)	A (293)	B (340)	C (351)
Fuzzy(0,0.5,1)	293	339	351
Fuzzy(0,1,1)	293	339	351
Crisp	286	319	320
Fuzzy(0,0.5,1)×5	152	157	170
Fuzzy(0,1,1)×5	113	121	128
Crisp×5	89	95	93

B. Results

The results we obtained are listed in Table III below. An explanation of the notation used is as follows:

- Fuzzy(0,0.5,1) A fuzzy set of a triangular shape where 0.5 is the ratio used for the centre value, to half the value given, and 1 is the ratio used for the upper value, i.e. value provided is used as an upper-limit.
- Fuzzy(0,1,1) A fuzzy set of a triangular shape where 1 is the ratio used for the centre value of the processing time, to half the value given, and 1 is the ratio used for its upper value, i.e. value provided is used as an upper-limit.
- Crisp Processing times are crisp values.
- Fuzzy(0,0.5,1)×5 Same as Fuzzy(0,0.5,1), but processing times are multiplied by 5.
- Fuzzy(0,1,1)×5 Same as Fuzzy(0,1,1), but processing times are multiplied by 5.
- Crisp×5 Same as Crisp, but values are multiplied by 5.

As a reminder, the higher the number of scheduled jobs, the better the algorithm is deemed to be.

The results obtained show that the best performance is obtained when using the fuzzy processing times in the optimistic form, using the representation of Fuzzy (0, 0.5,1) (i.e. jobs are most likely to finish in half of the upper value); the next best is the pessimistic fuzzy form, using the representation of Fuzzy(0,1,1), in which jobs are more likely to finish closer to the expected date.

This is to be expected as processors are freed quicker using the fuzzy times than when using crisp values. This is due to the fact that, when using crisp representation, jobs often hold the processors for longer than really needed, causing other jobs to become tardy and therefore unscheduled.

The difference between the performance of the above three methods becomes more acute when we have used the enlarged fuzzy sets. One can then see how we obtained the highest number of scheduled jobs, when using the optimistic representation of (0, 0.5,1), this was then followed by the less optimistic representation of (0,1,1), while the crisp schedulers came last in all three case studies.

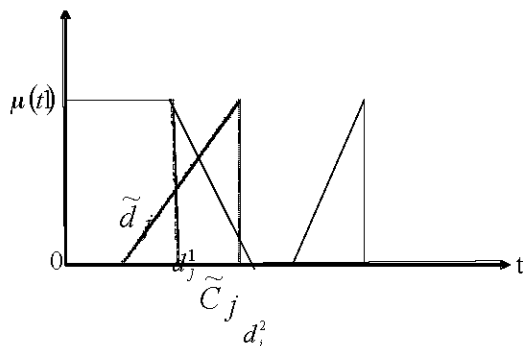


Fig. 7. Fuzzy due date with added tolerance

Finally, we also experimented by adding some tolerance to the due dates, so instead of using crisp upper values, we added a 10% tolerance, as shown in Fig. 7. While the added tolerance decreased the ratio of tardiness, by increasing the area of intersection between the two fuzzy sets, \tilde{C}_i and \tilde{d}_i , the number of scheduled versus unscheduled jobs was not largely affected and remained close to that obtained when using a crisp set representation.

VII. DISCUSSION

This paper has studied the case of a Grid scheduling problem where processing times were uncertain values. Fuzzy sets have been used to model the uncertain processing times and tabu search was used to search for the best solution that would maximize the number of scheduled jobs. Completion times were also fuzzy and the satisfaction criterion was consequently modelled by utilising fuzzy techniques. This research work investigated the implications of the fuzzification of an already working crisp system. The shape of fuzzy sets and their sizes were also changed. Informal observation has noted that jobs often fail almost immediately (caused for example by an incorrect command-line execution specification), but that, if not, then they will run for close to the pre-specified job length. This behaviour is not adequately captured by using simple triangles for fuzzy job processing times. Hence, we would like to explore different shapes of the fuzzy set representing processing time, perhaps using less conventional ones (non-triangular). Further investigations of the benefits of utilising fuzzy scheduling models in the context of scheduling service level agreements for the Grid will be need to be carried out in more detailed and realistic case studies. See [16] for more details of the use of fuzzy sets in the combinatorial optimisation of scheduling problems.

ACKNOWLEDGMENT

This work was funded by EPSRC through the Fundamental Computer Science for e-Science initiative (grant GR/S67661/01). The authors thank Viktor Yarmolenko and Rizos Sakellariou of the School of Computer Science, University of Manchester for their contribution to the ideas within this paper through their collaboration under EPSRC grant GR/S67654/01, and for supplying the datasets.

REFERENCES

- [1] I. Foster and C. Kesselman, Eds., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufman Publishers, 1998.
- [2] D. Ouelhadj, J.M. Garibaldi, J. MacLaren, R. Sakellariou, and K. Krishnakumar, "A multi-agent infrastructure and a service level agreement negotiation protocol for robust scheduling in grid computing," in *Proceedings of the European Grid Conference*, Lecture Notes in Computer Science, Springer-Verlag, 2005.
- [3] A. Abraham, R. Buyya, and B. Nath, "Nature's heuristics for scheduling jobs on computational grids," in *The 8th International Conference on Advanced Computing and Communications*, Cochin, India, 2000.
- [4] A. Yarkhan and J. J. Dongarra, "Experiments with scheduling using simulated annealing in a grid environment," in *Proceedings of the Third International Workshop on Grid Computing*, Lecture Notes In Computer Science, pp. 232–242. Springer-Verlag, 2002.
- [5] J. Cao and S. Jarvis, "Arms: An agent-based resource management system for grid computing," *Scientific Programming*, vol. 10, pp. 135–148, 2002.
- [6] S. A. Jarvis, D. P. Spooner, H. N. Lim Choi Keung, G. R. Nudd, J. Cao, and S. Saini, "Performance prediction and its use in parallel and distributed computing systems," in *Proceedings of the IEEE/ACM International Workshop on Performance Modelling, Evaluation and Optimization of Parallel and Distributed Systems*, Nice, France, 2003.
- [7] T. Braun, H. J. Siegel, N. Beck, D. Hensgen, and R. F. Freund, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 61, pp. 810–837, 2001.
- [8] H. Zhao and R. Sakellariou, "A low-cost rescheduling policy for dependent tasks on grid computing systems," in *Proceedings of the 2nd Across Grids Conference*, 2004.
- [9] C. Fayad and S. Petrovic, "A genetic algorithm for the real world fuzzy job-shop scheduling," in *Proceedings of The International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA/AIE-2005*, 2005, Lecture Notes in Computer Science, Springer-Verlag.
- [10] G. Klir and T. Folger, *Fuzzy Sets, Uncertainty and Information*, Prentice Hall, New Jersey, 1988.
- [11] F. Glover, "Heuristics for integer programming using surrogate constraints," *Decision Sciences*, vol. 8, pp. 156–166, 1977.
- [12] F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, 1997.
- [13] P. Brucker, *Scheduling Algorithms*, Springer, Berlin, 2nd edition, 1998.
- [14] D. Ouelhadj, J. M. Garibaldi, and R. Sakellariou, "A service level agreement protocol and tabu search for scheduling on computational grids," in *Proceedings of MISTA 2005: The 2ND Multidisciplinary Conference on Scheduling: Theory and Applications*, NY, USA, 2005.
- [15] M. Sakawa and R. Kubota, "Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy due date through genetic algorithms," *European Journal of Operational Research*, vol. 120, no. 2, pp. 393–407, 2000.
- [16] Roman Slowinski, *Scheduling under Fuzziness*, Physica-Verlag, 2000.