Hybrid Algorithms for Independent Batch Scheduling in Grids

Fatos Xhafa*

Technical University of Catalonia, Spain
E-mail: fatos@lsi.upc.edu
**Corresponding author*

Joanna Kołodziej

University of Bielsko-Biała, Poland
E-mail: jkolodziej@ath.bielsko.pl

Leonard Barolli

Fukuoka Institute of Technology, Japan
E-mail: barolli@fit.ac.jp

Vladi Kolici, Rozeta Miho

Polytechnic University of Tirana, Albania
E-mail: {vkolici,rmiho}@fti.upt.al

Makoto Takizawa

Seikei University, Kichi-Joji, Tokyo, Japan
E-mail: makoto.takizawa@ieee.org

Abstract:

Grid computing has emerged as a wide area distributed paradigm for solving the large-scale problems in science, engineering, etc., known as the family of eScience Grid-enabled applications. Computing efficiently a planning of incoming jobs to available machines in the Grid system is a main requirement for optimized system performance. One version of the problem is that of independent batch scheduling in which jobs are assumed independent and are scheduled in batches aiming to minimize the makespan and flowtime. Given the hardness of the problem, heuristics are used to find high quality solutions for practical purposes of designing efficient Grid schedulers. Recently, considerable efforts are done in implementing and evaluating not only stand alone heuristics and metaheuristics but also their hybridization into even higher level algorithms. In this paper we present a study on the performance of two popular algorithms for the problem, namely Genetic Algorithms (GAs) and Tabu Search (TS), and two hybridizations of them, namely, the GA(TS) and GA-TS which differ in the way the main control and cooperation among GA and TS are implemented. The hierarchic and simultaneous opti-

The final publication is available at: http://dx.doi.org/10.1504/IJWGS.2012.048402

mization modes are considered for the bi-objective scheduling problem. The evaluation is done using different grid scenarios generated by a grid simulator. The computational results showed that the hybrid algorithms outperforms both the GA and TS for makespan parameter but not for the flowtime parameter.

Keywords: Computational Grids, Meta-heuristics, Genetic Algorithms, Tabu Search, Hybridization, Makespan, Flowtime, Hierarchic Optimization, Simultaneous Optimization.

Reference to this paper should be made as follows: Xhafa, F., Kołodziej, J., Barolli L., Kolici, V, Miho, R. and Takizawa, M. (xxxx) 'Evaluation of Hybrid Algorithms for Independent Batch Scheduling in Computational Grids', *Int. J. Web and Grid Services*, Vol. x, No. x, pp.xxx–xxx.

Biographical Notes:

Fatos Xhafa holds a permanent position of *Professor Titular* at the *Departament of Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, UPC* (Spain). He was a Visiting Professor at the Department of Computer Science and Information Systems, Birkbeck, University of London, UK (2009/2010) and a Research Associate at College of Information Science and Technology, Drexel University, Philadelphia, USA (2004/2005). His research interests include parallel and distributed algorithms, combinatorial optimization, approximation and meta-heuristics, networking and distributed computing, Grid and P2P computing. He has widely published in peer reviewed international journals, conferences/workshops, book chapters and edited books and proceedings in the field and is Editor in Chief of the International Journal of Space-based and Situated Computing, and of International Journal of Grid and Utility Computing, Inderscience Pubs. He is also actively participating in the organization of several international conferences in the field.

Joanna Kołodziej graduated in Mathematics from the Jagiellonian University in Krakow in 1992, where she also obtained the PhD in Computer Science in 2004. She joined the Department of Mathematics and Computer Science of the University of Bielsko-Biała as an Assistant Professor in 1997. She has served and is currently serving as PC Co-Chair, General Co-Chair and IPC member of several international conferences and workshops including PPSN 2010, ECMS 2011, CISIS 2011, 3PGCIC 2011, CISSE 2006, CEC 2008, IACS 2008-2009, ICAART 2009-2010. Dr Kołodziej is Managing Editor of IJSSC Journal and serves as a EB member and guest editor of several peer-reviewed international journals.

Leonard Barolli received BE and PhD degrees from Tirana University and Yamagata University in 1989 and 1997, respectively. From April 1997 to March 1999, he was a JSPS Post Doctor Fellow Researcher at Department of Electrical and Information Engineering, Yamagata University. From April 1999 to March 2002, he worked as a Research Associate at the Department of Public Policy and Social Studies, Yamagata University. From April 2002 to March 2003, he was an Assistant Professor at Department of Computer Science, Saitama Institute of Technology (SIT). From April 2003 to March 2005, he was an Associate Professor and presently is a Full Professor, at Department of Information and Communication Engineering, Fukuoka Institute of Technology (FIT). Dr. Barolli has published about 300 papers in referred Journals, Books and International Conference proceedings. He was an Editor of the IPSJ Journal and has served as a Guest Editor for many

International Journals. Dr. Barolli has been a PC Member of many International Conferences and was the PC Chair of IEEE AINA-2004 and IEEE ICPADS-2005. He was General Co-Chair of IEEE AINA- 2006 and AINA-2008, Workshops Chair of iiWAS-2006/MoMM-2006 and iiWAS-2007/MoMM- 2007, Workshop Co-Chair of ARES-2007, ARES-2008, IEEE AINA-2007 and ICPP-2009. Presently, he is General Co-Chair of CISIS-2010 and IEEE AINA-2010. Dr. Barolli is the Steering Committee Chair of CISIS International Conference and is serving as Steering Committee Co-Chair of IEEE AINA and NBiS International Conferences. He is organizers of many International Workshops. His research interests include high-speed networks, Grid computing, P2P, ad-hoc and sensor networks. He is member of IEEE, IPSJ and SOFT.

Vladi Kolici received his B.S and M.S degrees in Telecommunication Engineering from Polytechnic University of Tirana (PUT) in 1997 and 2005, respectively. He obtained his Ph.D from PUT in May 2009. From 1997 to 2004, he was a Research Associate and from 2005 to present he is a Lecturer at Department of Electronics and Telecommunications, Faculty of Information Technology, PUT. He is teaching several courses in the areas of wireless and mobile networking, P2P systems and quality of services. Dr. Kolici has published several papers in International and National Conference Proceedings in the areas of P2P and Ad-Hoc networks. His research interests include P2P networks, wireless and mobile networks, and high speed networks.

Rozeta Miho received her B.S and M.S degrees in Electronic and Telecommunication Engineering from Polytechnic University of Tirana (PUT), Albania, in 1985 and 1989, respectively. She obtained her Ph.D from PUT in November 1995. From 1985 to 1995, she was a lecturer, from 1996 to 2001 Assistant Professor, from 2001 to 2007 Associate Professor, and presently she is a Full Professor of PUT. From May 2009, she is the Dean of Faculty of Information Technology, PUT. She is teaching several courses in the areas of telecommunication networks, optical communications and optical fibre networks. Prof. Miho has published several papers in International and National Conference Proceedings in the areas of optical WDM networks, P2P and Ad-Hoc networks. Her research interests include P2P networks, optical networks, wireless networks and high speed networks.

Makoto Takizawa is a Professor at the Seikei University, Japan...TBC

1 Introduction

Grid computing has emerged as a wide area distributed paradigm for solving the large-scale problems in science, engineering, etc., known as the family of eScience Grid-enabled applications (Constantini et al., 2010) as well for advanced services (Flahive et al., 2009). Computational Grid involves the combination of many computing resources into a network for the execution of computational tasks. The resources are distributed across multiple organizations, administrative domains having their own access, usage policies and local schedulers. The tasks scheduling and the effective management of the resources in such systems are complex and therefore, demands sophisticated tools for analyzing the algorithms performances before applying them to the real systems (Taniar et al., 2007; Goel et al., 2005; Tanaka et al., 2008; Santos et al., 2008; Reinhard et al., 2008; Taniar et al., 2008).

During the past three decades, meta-heuristics have been among most studied approaches to efficiently solve combinatorial optimization problems. Families of meta-heuristics such as local search methods, population-based methods and biologically inspired methods were developed for most computationally hard problems. More recently, attention has been shifted to the design and implementation of high level algorithms that combine heuristics methods. These algorithms, known as hybrid algorithms, or hybrid meta-heuristics for the case of meta-heuristics being hybridized, aim to explore the existing synergies among stand-alone heuristics methods in order to achieve even more efficient and robust algorithms. Based on this premise, many optimization frameworks have been proposed in the field.

One of the motivations for developing hybrid meta-heuristics is that of coping in practice with dynamic and large instances from real-world problems, that is, a clear practical relevance. In this context, we are interested to see how hybridization could help in efficiently solving the scheduling problem in Computational Grids. The problem is very complex due to the large-scale, heterogeneous and dynamic nature of Grid systems. Additionally, the problem can be formulated for different modes such as immediate and batch mode and is multi-objective in its general formulation.

Various classes of search algorithms can be considered for the purposes of hybridization, such as exact methods, simple heuristic methods (*ad hoc* methods) and meta-heuristics. Moreover, meta-heuristics themselves are classified into local search based methods, such as Simulated Annealing, Tabu Search, Variable Neighborhood Search, etc., population based methods such as Genetic Algorithms, Memetic Algorithms, etc., and other classes of nature inspired meta-heuristics such as Particle Swarm, Ant Colony Optimization, etc. Therefore, in principle, one has many choices to select and combine different heuristic methods either methods of the same type (e.g. local search methods) or methods of different types (e.g. population based methods with local search methods). Thus, as there are plenty of meta-heuristics methods, which of them coming with many variations, the questions that arise for hybridizations essentially are "*how to hybridize?*" and "*where to hybridize*"? The former questions refers how to select and combine the different methods, in some sense one needs to break the arbitrariness on *how* to select methods or modules for hybridization. The later refers to the fact that once we select the methods to be hybridized, *where* to introduce the combination, e.g. in terms of algorithm flow. Formalizing of hybridization approaches is rather difficult to embrace all kinds of hybridizations (Talbi, 2002; Jourdan et al., 2009). For many, hybridization is seen as a way to solve real-world problems otherwise intractable with stand alone heuristic methods. Fortunately, the prototyping of hybrid meta-heuristics has become easy due to the support of many libraries Alba et al. (2006); Cahon et al. (2004); Lau et al. (2004) proposed in the field.

Similarly as for other combinatorial optimization problems, many heuristic approaches have been proposed in the literature for the problem of Grid scheduling Abraham et al. (2000); Ritchie and Levine (2003); Xhafa (2007); Xhafa et al. (2007, 2009). The existing approaches include local search methods, population based methods, etc. Therefore, such heuristics methods are candidates for hybrid approaches.

In our previous works Xhafa et al. (2009, 2011) we presented two hybrid algorithms, GA(TS) and GA-TS, for the problem of independent batch scheduling in

Grid systems. In the former, the flow of the hybrid algorithm was that of a Genetic Algorithm (GA) in which Tabu Search (TS) was used to make locally improvements of new individuals of the population. In the later, the GA and TS were executed in a sequence-like manner, first the GA, next TS on the best output solution of GA. In both versions, the independent batch scheduling was formulated as a bi-objective optimization problem and the *hierarchic* optimization with makespan as primary objective and flowtime as secondary objective was considered.

In this work, we evaluate the GA(TS) and GA-TS hybrid algorithms for the bi-optimization model in which both makespan and flowtime are optimized simultaneously using a weighted sum of the objectives as a single objective function. The proposed algorithm has been experimentally evaluated using Grid simulator and the results are compared with the results achieved by both GAs and TS used as stand-alone heuristic schedulers. For the evaluation we have considered different sizes of the problem instance (number of tasks and number of machines in the Grid system). For the evaluation of results, we have used as well the results previously reported in Xhafa et al. (2009, 2011)

The rest of the paper is organized as follows. In Section 2, we briefly present the scheduling of independent tasks considered as a bi-objective optimization problem in this work. In Section 3, different types of hybridizations are presented. The GAs and TS for the problem as well as the GA-TS hybrid approach are given in Section 4. The experimental study and some computational results are given in Section 5. We conclude in Section 6 with some remarks and indications for future work.

2 Scheduling of independent tasks in computational grids

Computational Grids are parallel in nature. Remote users can connect to the Grid systems and independently submit tasks or applications to the system which should be scheduled for execution in Grid nodes. It is in this context where arises the independent task scheduling, in which there are no dependencies among the tasks. Within this context, depending on user requirements one can consider the immediate mode (tasks or applications are considered for allocation as soon as they enter the system) or batch mode (tasks or applications are grouped into batches and scheduled). The later case of batch mode frequently arises for the case of periodic submissions.

2.1 Independent batch scheduling

In this work we are interested in scheduling of independent tasks to grid resources. The formal definition of the problem is based on the definition of the Expected Time to Compute (ETC) matrix in which $ETC[j][m]$ indicates an estimation of the completion of task j in resource m. In fact, one possible way to compute the entries $ETC[j][m]$ is to divide the workload of task j by the computing capacity of resource m. Under the ETC matrix model, the scheduling problem specification is as follows:

- A number of independent *tasks* to be allocated to grid resources. Each task has to be processed entirely in a single resource and is not preempted (once started, a task runs until completion).
- A number of *machines* that are candidate to participate in the allocation of tasks.
- The *workload* (in millions of instructions) of each task.
- The *computing capacity* of each machine (in *Mips*).
- The ready times, denoted $ready_m$, indicating when machine m will have finished the previously assigned tasks. This parameter measures the previous workload of a machine.
- The ETC matrix of size $nb\_tasks \times nb\_machines$, where $ETC[j][m]$ is the value of the expected time to compute of task j in machine m.

2.2 Optimization criteria

The quality of a schedule can be measured using several optimization criteria, such as minimizing the *makespan* (that is, the finishing time of the latest task), the *flowtime* (i.e., the sum of finalization times of all the tasks), the *completion time* of tasks in every machine (closely related to makespan), which formally can be defined as follows:

- *makespan*: $\min_{S_i \in Sched}\{\max_{j \in Tasks} F_j\}$ and,
- *flowtime*: $\min_{S_i \in Sched}\{\sum_{j \in Tasks} F_j\}$,

where F_j denotes the time when task j finalizes and $Sched$ is the set of all possible schedules. In this work we consider two types of optimization models, namely the hierarchic and simultaneous approach. In the former, we assume that the makespan is a privileged criterion and flowtime is the second less important scheduler performance measure. This hierarchy in the criteria importance is a basis for designing a hierarchical optimization algorithm, in which the makespan's values cannot be worsened when optimizing the flowtime. In the simultaneous approach, both objectives are optimized simultaneously in a weighted sum objective function: *min* $\lambda \cdot makespan + (1 - \lambda) \cdot mean\_flowtime$, for $\lambda = 0.75$.

2.3 Batch mode scheduling

Processing tasks that arrive to Grid systems in batch mode is one of the most common scenarios in Grid systems. It arises due to the independent submissions of tasks by many user geographically distributed. The problem also arises in massive processing where tasks spawned by an application do not have dependencies among them. The batch Grid scheduling essentially includes the phases below, and are shown in Fig. 1.

1. Gather the information on available resources (machine pool)

2. Gather the information on pending jobs (job pool)
3. Make a batch and *compute a planning for that batch*
4. Allocate jobs
5. Monitor job execution (failed jobs are re-scheduled again, entering pool job).

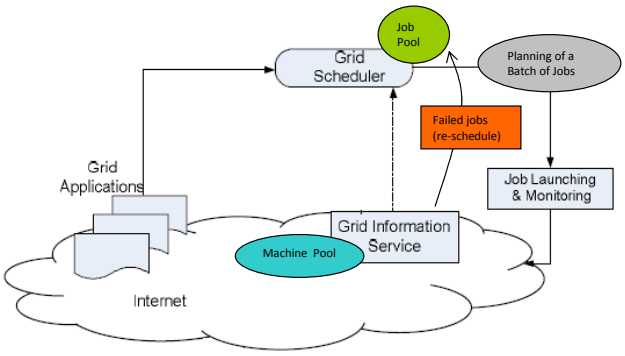

Figure 1 Batch scheduling.

Dealing with dynamic nature of Grid systems. One important requirements in Grid systems is to address the dynamics of the systems, such as machine failure, task failure, etc., due to which, depending on scheduling policy, tasks must be migrated to other nodes on the system or should be re-scheduled. Batch scheduling has the limitation that tasks are not scheduled for processing as soon as they enter the system. However, we cope with the dynamics of Grid system by using batch processing in very short intervals of time. That is, we keep the time among two successive batches very small (e.g. less than 100 seconds) so that changes in the system are unlikely to happen in such short time. On the other hand, using the batch processing has the advantage that an optimized planning of tasks to machines can be computed as compared to immediate mode processing.

3 Hybridization of meta-heuristics

3.1 How to hybridize, where to hybridize?

One main issue in designing hybrid algorithms is the high degree of arbitrariness, that is, the many ways one can choose to combine different resolution methods. Said in other words, the questions are how to hybridize, where to hybridize? The *how* refers to the way we "modify"/"combine" some parts of a meta-heuristics by using other meta-heuristics resulting in a new control flow. The *where* refers to the fact that hybridization can take place at different phases of the meta-heuristics, starting with the computation of the initial solutions up to the modification of some procedures of the original heuristic method by using procedures of other heuristic methods. For instance, such procedures are neighborhood exploration or genetic operators.

Let us consider for example a Genetic Algorithm. How to hybridize with local search methods? We could use the GA as a main algorithm and call local search methods along the flow. Where to hybridize? Local search methods can be used

at different places along the GA flow: to generate some of the individuals of the first population and thus introduce more diversity among individuals, to implement some genetic operators (e.g. mutation), improving offsprings by local search, etc.

3.2 Hybridization "recipe"

In order to design a hybrid algorithm one essentially needs to specify the information below:

1. Number of methods to hybridize;
2. Methods to hybridize;
3. Level of hybridization.

The first refers to the number of methods to hybridize. In principle, there is no limitations on how many heuristics should be hybridized, however, most proposed approaches in literature consider two-three heuristics. Then, one has to provide the concrete methods to hybridize by choosing from the available heuristics for the problem (exact, local search, population-based, etc.) Finally, and most importantly, one has to design the flow of the new hybrid algorithm. For this, the crucial point is the level of hybridization, which refers to the degree of coupling between the meta-heuristics, the execution sequence and the control strategy.

3.3 Level of hybridization

The level of hybridization expresses the dependencies among the flows of the considered heuristics. High level hybrid heuristics are loosely coupled and low level hybrid heuristics are strongly coupled, as described next.

In the *Loosely coupled* case the hybridized meta-heuristics preserve their identity, namely, their flow is fully used in the hybridization. This case is also referred to as *high level of hybridization* and can be seen as a chain of meta-heuristics executions

$$MH_1 \rightarrow MH_2 \rightarrow \cdots \rightarrow MH_k$$

which can be further looped a certain number of iterations or until a stopping condition is met. The meaning is that first we run MH_1; the output solution is passed on to MH_2 and so on. The best found solution by MH_k is the final solution. In this case, the notation $MH_1 + MH_2 + \cdots + MH_k$ is used (in our case GA-TS).

In *Strongly coupled* hybridization the combined meta-heuristics interchange their inner procedures, resulting in a *low level of hybridization*. The level of hybridization expresses the degree of interaction among the meta-heuristic components in the hybrid structure. In this case, usually one of the heuristics is the main algorithm, which during its flow calls other heuristics procedures. For instance, we can run GA and then additionally to crossover and mutation, we can apply TS for improving the newly generated solutions. The notation $MH_1(MH_2)$ is used in this case to express that MH_1 is the main algorithm and MH_2 is subordinated to that (GA(TS), in our case).

Execution sequence. The execution sequence refers to the computing medium, being sequential or distributed. In the *sequential* case the meta-heuristics flows are run sequentially while in *parallel* setting the meta-heuristics flows are run in parallel in a networked computing environment.

Control strategy The control strategy refers to a flow implemented by the hybrid meta-heuristic. The control can be

- *Coercive*: the main flow is that of one of the meta-heuristics, the other meta-heuristics flow is subordinated to the main flow. It should be noted that in this case, the implementation of the heuristics requires fine grain implementation of the search procedures in a way that it's easy to get the state of the search at any time along the search process.

- *Cooperative*: the meta-heuristics explore the solution space cooperatively (eventually, they can explore different parts of the solution space.)

4 Hybridization of GA and TS algorithms

4.1 The GA(TS) hybrid algorithm

For the design of our hybrid approach we consider two well-known meta-heuristics: Genetic Algorithms (GAs) and Tabu Search (TS). Both GAs and TS have been developed for the independent task scheduling in Xhafa et al. Xhafa et al. (2007) and Xhafa et al. (2009) in sequential setting. We have considered the Steady-State GA in this work. The choice of these two meta-heuristics is based on the following observations. First, grid schedulers should be very fast in order to adapt to dynamic nature of computational grids. Therefore, a fast convergence of the main algorithm is preferable in this case, which can be achieved through a good trade-off between exploration and exploitation of the search. Second, in order to achieve high quality planning in a very short time, it is suggestive to combine the *exploration* of the solution space by a population of individuals with the *exploitation* of neighbourhoods of solutions through local search. In such case, GAs and TS are among the best representatives of population based and local search methods, respectively.

We are thus considering the case of hybridization of two meta-heuristics running in sequential environment. We have considered a low level hybridization and the coercive control strategy. Roughly, our hybrid algorithm runs the GA as the main algorithm and calls TS to improve individuals of the population.

The hybridization scheme is shown in Figure 2. It should be noted that in the hybridization scheme in Figure 2, instead of replacing the mutation procedure of GAs by the TS procedure, we have added a new function to the GA Population class (namely `apply_TabuSearch`) for applying the TS. This new function could be applied to any individual of the current population, however, this is computationally costly. In our case, given that we want to run the Grid scheduler in short times, the `apply_TabuSearch` is applied with small probability[a]. In fact, this parameter can well be used to tune the convergence of the GA since TS usually provides substantial improvements to individuals.

[a]This is a user input parameter. `apply_TabuSearch` is applied roughly to 30% of individuals

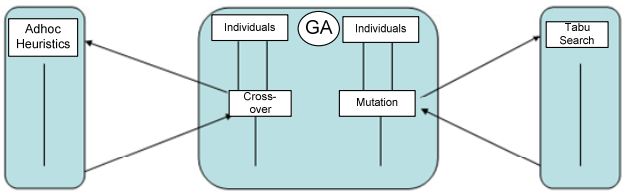

Figure 2 The hybrid GA(TS) scheme.

4.2 The GA-TS hybrid approach

The specification of the hybridization for GA-TS is as follows:

- Two meta-heuristics are hybridized.
- The selected methods are GA and TS.
- The hybridization is loosely coupled, in the sequence GA is the first heuristic to run and TS is the second one.
- The implementation is for sequential setting environments.

The hybridization scheme is shown in Figure 3.

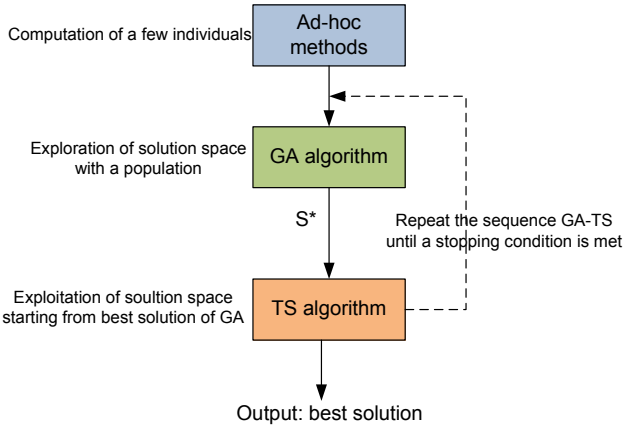

Figure 3 The hybrid GA-TS scheme.

In the hybridization scheme in Fig. 3, initially, we generate a few individuals using *ad hoc* methods aiming to achieve and keep diversity of population in GA. Then, GA is activated, which outputs a best solution S^* upon meeting a stopping condition. The best solution S^* is then passed on to the TS algorithm in input (as starting solution). The hybridization scheme can iterate the sequence GA→TS an *a priori* number of times or until a stopping criteria is met. It should be also noted that the stopping condition for each heuristic needs not to be the same. For instance, in GA the stopping condition could be number of generations while in TS could be a maximum execution search time.

We shortly present next both the GA and TS meta-heuristics for independent task scheduling in computational grids (refer to Xhafa et al. (2007) and Xhafa et al. (2009) for details.)

4.3 GAs for the scheduling problem in Grids

We have implemented the Steady State version of GAs for the purpose of this work. In Steady State GAs, a few good individuals of population are selected and crossed. Then, the worst individuals of the population are replaced by the newly generated descendants; the rest of the individuals of the population survive and pass to the next generation (see Alg. 1).

Algorithm 1 Genetic Algorithm template

```
Generate the initial population P^0 of size μ;
Evaluate P^0;
while not termination-condition do
  Select the parental pool T^t of size λ; T^t := Select(P^t);
  Perform crossover procedure on pairs of individuals in T^t
  with probability p_c; P_c^t := Cross(T^t);
  Perform mutation procedure on individuals in P_c^t
  with probability p_m; P_m^t := Mutate(P_c^t);
  Evaluate P_m^t ;
  Create a new population P^{t+1} of size μ from individuals in P^t and/or P_m^t ;
  P^{t+1} := Replace(P^t; P_m^t)
  t := t + 1;
end while
return Best found individual as solution;
```

The genetic operators and methods used in the implementation are as follows:

- *Initialization methods* are MCT and LJFR-SJFR implemented in Xhafa et al. Xhafa et al. (2007,?);
- *Selection operator*: Linear ranking;
- *Crossover operator*: Cycle Crossover (CX);
- *Mutation operator*: Mutate Rebalancing;

The values for the rest of parameters are given in Section 5.

4.4 Tabu Search for the scheduling problem in Grids

Tabu Search (TS) has shown its effectiveness in a broad range of combinatorial optimization problems and distinguishes for its flexibility in exploiting domain/problem knowledge (see Alg. 2).

The main procedures used in TS are summarized next:

- *Initial solution* is found using Min-Min method Xhafa et al. (2007).
- *Historical memory*: Both short and long term memories have been used in TS algorithm. For the *recency* memory, a matrix TL ($nb\_tasks \times nb\_machines$) is used to maintain the tabu status. In addition, a tabu hash table (TH) is maintained in order to further filter the tabu solutions.
- *Movement*: Two types of movement are used, namely, transfer (moves a task from a machine to another one, appropriately chosen) and swap (two tasks assigned to two different machines are swapped). The neighborhood exploration is done using a steepest descent - mildest ascent method.

Algorithm 2 Tabu Search Algorithm

```
begin
Compute an initial solution s;
let ŝ ← s;
Reset the tabu and aspiration conditions;
while not termination-condition do
  Generate a subset N*(s) ⊆ N(s) of solutions such that:
       (none of the tabu conditions is violated) or (the aspiration criteria hold)
  Choose the best s' ∈ N*(s) with respect to the cost function;
  s ← s';
  if improvement(s', ŝ)) then
    ŝ ← s';
  end if
  Update the recency and frequency;
  if (intensification condition) then
    Perform intensification procedure;
  end if
  if (diversification condition) then
    Perform diversification procedures;
  end if
end while
return ŝ;
end;
```

- *Aspiration criteria*: Several aspiration criteria are used to remove the tabu status of movements. They are defined using the fitness of solutions as well as information from recency matrix.
- *Intensification*: Implemented using elite solutions;
- *Soft Diversification*: Implemented using penalties to ETC values, task distribution and task freezing.
- *Strong Diversification*: Implemented using large perturbations of solutions.

The concrete values for the parameters of TS are given in Section 5.

5 Experimental study

We have used the HyperSim-G simulator (see Fig. 4), a Grid simulator Xhafa et al. (2007), to evaluate our GA-TS hybrid algorithm. HyperSim-G extends HyperSim simulation package, an open source, general-purpose discrete event simulation library developed in C++.

5.1 Simulation environment setting

For the evaluation of the GA-TS hybrid algorithm we used the HyperSim-G Grid simulator Xhafa et al. (2007). We considered three Grid scenarios: small, medium and large sizes with 32 hosts / 521 machines, 64 hosts / 1024 machines,

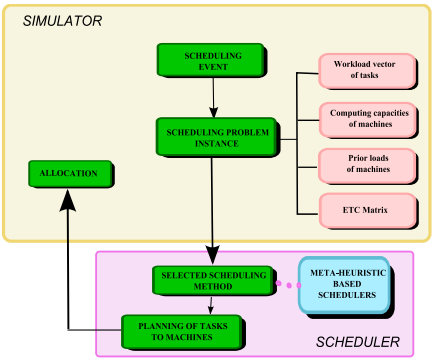

Figure 4 General Flowchart of HyperSim-G Simulator.

and 128 hosts / 2048 machines respectively. We also consider static and dynamic grid environments, as explained next.

Simulator's configuration for the static case

In the static grid the number of tasks and machines are kept constant during the execution of the simulator. It is thus assumed that the system starts with a certain number of tasks and machines and that there are no machine drops from the system. The configuration of simulator is defined by parameters presented in Table 1.

Table 1 Simulators' static case configuration.

	Small	Medium	Large
Init./Total hosts	32	64	128
Mips	n(1000, 175)		
Init./Total tasks	512	1024	2048
Workload	n(250000000, 43750000)		
Host selection	All		
Task selection	All		
Local policy	SPTF		
Number of runs	30		

Simulator's configuration for the dynamic case

In the dynamic case the numbers of tasks and machines can vary over time according to the probability distributions specified in the simulator. More precisely, one the one hand new tasks can enter the system, and on the other, machines can leave the Grid system, which provokes the tasks assigned to those machines to be re-scheduled. The configuration of simulator is presented in Table 2.

5.2 Parameter setting of GA and TS algorithms

For the GA algorithm we have used the parameter values given in Table 3 and for TS those given in Table 4.

5.3 Static case: Computational results and evaluation

The simulator is run 30 times for each scenario and computational results for makespan and flowtime are averaged. Standard deviation (at 95% confidence inter-

Table 2 Simulators' dynamic case configuration.

	Small	Medium	Large
Init. hosts	32	64	128
Max. Hosts	37	70	135
Min. Hosts	27	58	121
Mips	n(1000, 175)		
Add host	n(625000,93750)	n(562500,84375)	n(500000,75000)
Delete host	n(625000,93750)		
Total tasks	512	1024	2048
Init. tasks	384	768	1536
Workload	n(250000000, 43750000)		
Interrarival	e(7812.5)	e(3906.25)	e(1953.125)
Activation	resource_and_time_interval(250000)		
Reschedule	True		
Host select	All		
Task select	All		
Local policy	Sptf		
Number runs	15		

Table 3 Parameter values of GA.

Parameter	**Value**
evolution steps	$20 * nb\_tasks$
population size	$4 * (\log_2(nb\_tasks) - 1)$
intermediate population size	$(population\ size)/3$
cross probability	1.0
mutation probability	0.4

Table 4 Parameter values of TS.

Parameter	**Value**
number of iterations	$nb\_tasks * nb\_machines$
max. tabu status	$1.5 * nb\_machnies$
number of repetitions before activating intensification/ diversification	$4 * \ln(nb\_tasks) * \ln(nb\_machines)$
number of iterations per intensification/diversification	$\log_2(nb\_tasks)$
number of iterations for aspiration criteria	$max\_tabu\_status/2 - \log_2(max\_tabu\_status)$

val) is also reported. The results for makespan and flowtime are given in Tables 5 and 6, for GA, TS, GA(TS) and GA-TS in both hierarchic and simultaneous optimization mode.

Table 5 Makespan values in static case.

	Small	Medium	Large
GA hierarchic	2808662.116	2760024.390	2764455.222
	±1,795%	±1,010%	±0,745%
GA simultaneous	2809174.783	2762104.889	2764449.781
	±1,791%	±1,008%	±0,726%
TS hierarchic	2805531.301	2752355.018	2748878.934
	±1,829%	±1,056%	±0,669%
TS simultaneous	2823526.492	2790908.892	2825514.732
	±1,741%	±1,012%	±0,804%
GA-TS hierarchic	2805527.381	2752256.136	**2746662.875**
	±1,829%	±1,056%	±0,675%
GA-TS simultaneous	2808912.321	2762317.234	2769866.106
	±1,797%	±1,004%	±0,750%
GA(TS) hierarchic	**2805519.428**	**2751989.166**	2812776.300
	±1,829%	±1,058%	±1,176%
GA(TS) simultaneous	2905693.092	2875899.153	2879641.808
	±1,995%	±1,191%	±0,819%

Table 6 Flowtime values in static case.

	Small	Medium	Large
GA hierarchic	709845463.699	1405493291.442	2811723598.025
	±1,209%	±0,655%	±0,487%
GA simultaneous	709228929.998	1403917124.018	**2809757248.263**
	±1,194%	±0,646%	±0,476%
TS hierarchic	710189541.278	1408001699.550	2812229021.221
	±1,124%	±0,616%	±0,455%
TS simultaneous	**707139704.477**	**1402094313.667**	2811842741.040
	±1,132%	±0,613%	±0,503%
GA-TS hierarchic	709649937.592	1406440866.567	2812036779.732
	±1,132%	±0,631%	±0,473%
GA-TS simultaneous	708928372.500	1403896743.275	2809768098.009
	±1,172%	±0,622%	±0,475%
GA(TS) hierarchic	711183944.069	1409127007.870	2811605453.116
	±1,174%	±0,604%	±0,465%
GA(TS) simultaneous	709647950.762	1408018070.562	2815647561.224
	±1,225%	±0,679%	±0,512%

As can be seen from Table 5, for makespan value the GA(TS) in its hierarchic version outperforms GA, TS and GA-TS for small and medium size instances, while GA-TS in its hierarchic version achieved the best results for large size grid scenarios. Overall, hybrid versions performed better that stand alone versions for makespan optimization. On the other hand, from Table 6, we can see that the stand alone approaches of GA and TS outperformed hybrid approaches for the flowtime optimization.

5.4 Dynamic case: Computational results and evaluation

In the dynamic case, the simulator is run 15 times for each scenario and computational results for makespan and flowtime are averaged. Standard deviation (at 95% confidence interval) is also reported. The results for makespan and flowtime are given in Table 7 and Table 8, resp. In the table, we present the results for GA (hierarchic), TS (hierarchic), GA(TS) (hierarchic) hybrid algorithm Xhafa et al. (2009).

Table 7 Makespan values in dynamic case.

	Small	Medium	Large
GA hierarchic	2982577.400	2893818.392	2841147.967
	±2,333%	±1,219%	±1,734%
GA simultaneous	2984736.895	2894982.335	2846359.851
	±2,214%	±1,375%	±1,726%
TS hierarchic	2979512.919	2888888.760	**2834225.002**
	±2,304%	±1,107%	±1,713%
TS simultaneous	2987369.099	2890788.084	2847143.199
	±2,508%	±1,245%	±1,634%
GA-TS hierarchic	2987369.099	2897935.151	2839885.306
	±2,418%	±1,038%	±1,711%
GA-TS simultaneous	2982582.663	2895996.446	2837073.631
	±2,348%	±1,330%	±1,761%
GA(TS) hierarchic	2973568.178	2889079.691	2847424.376
	±2,205%	±1,171%	±1,571%
GA(TS) simultaneous	**2964782.651**	**2886720.095**	2838111.918
	±2,353%	±1,060%	**±1,137%**

Table 8 Flowtime values in dynamic case.

	Small	Medium	Large
GA hierarchic	715400219.482	1410799837.874	2803579399.761
	1,524	0,699	0,731
GA simultaneous	714192855.967	1409918526.549	2800300798.758
	1,441	0,711	0,708
TS hierarchic	716241094.775	1409244604.279	2798012368.133
	1,377	0,638	0,685
TS simultaneous	**713046299.130**	**1406852379.473**	**2797549933.700**
	1,450	0,643	0,737
GA-TS hierarchic	715573207.887	1410988533.602	2800981497.939
	1,381	0,640	0,705
GA-TS simultaneous	714417906.975	1408474213.235	2799381277.911
	1,482	0,673	0,693
GA(TS) hierarchic	717402597.534	1413169482.528	2800824873.327
	1,223	0,638	0,677
GA(TS) simultaneous	714216794.135	1413274833.742	2801052281.191
	1,388	0,726	0,733

As can be seen from Table 7, for makespan value the GA(TS) algorithm (in its simultaneous version) performed better than GA, TS and GA-TS for all but large size instances. On the other hand, from Table 8, we can see that TS (in its simultaneous version) performed better than all other methods for flowtime value.

6 Conclusions and future work

In this paper we have presented an evaluation of two hybrid algorithms, namely GA(TS) and GA-TS, for the problem of independent batch scheduling in Grid systems under hierarchic and simultaneous optimization models. In the GA(TS), the flow of the hybrid algorithm was that of a Genetic Algorithm (GA) in which Tabu Search (TS) was used to make locally improvements of new individuals of the population. In GA-TS, the GA and TS were executed in a sequence-like manner, first the GA, next TS on the best output solution of GA. In both versions, the independent batch scheduling was formulated as a bi-objective optimization problem and the *hierarchic* optimization mode, with makespan as primary objective and flowtime as secondary objective, and *simultaneous* optimization using a weighted sum of the objectives as a single objective function, were considered. The proposed algorithms have been experimentally evaluated using Grid simulator and the results are compared with the results achieved by both GA and TS as stand-alone heuristic schedulers. For the evaluation we have considered different sizes of the problem instance (number of tasks and number of machines in the Grid system). The experimental study showed that hybrid versions performed best for the optimization of the makespan while stand alone versions of meta-heuristics performed better for the flowtime.

In our future work we plan to implement multi-objective versions of meta-heuristics such as Multi-Objective GAs (MOGAs), to compute the Pareto front, which would serve as a basis for a decision taking during the scheduling phase in Grid systems.

References

A. Abraham, R. Buyya, and B. Nath. Nature's heuristics for scheduling jobs on computational grids. In *The 8th IEEE International Conference on Advanced Computing and Communications*, India, 2000.

E. Alba, F. Almeida, M. Blesa, C. Cotta, M. Díaz, I. Dorta, J. Gabarró, C. León, G. Luque, J. Petit, C. Rodríguez, A. Rojas, and F. Xhafa. Efficient parallel LAN/WAN algorithms for optimization. the Mallba project. *Parallel Computing*, 32(5-6):415–440, 2006.

S. Cahon, N. Melab, and E. Talbi. Building with paradisEO reusable parallel and distributed evolutionary algorithms. *Parallel Computing* 30(5-6):677-697, 2004.

Costantini, A., Gutierrez, E., Cacheiro, L., J., Rodriguez, A., Gervasi, O., Lagana , A., On the extension of the grid-empowered molecular science simulator: MD and visualisation tools, *International Journal of Web and Grid Services (IJWGS)*, Year: 2010 Vol. 6, No. 2, pp. 141-159

A. Flahive, D. Taniar, W. Rahayu, B.O. Apduhan. Ontology tailoring in the Semantic Grid, *Computer Standards Interfaces*, 31(5): 870-885, 2009

S. Goel, H. Sharda, D. Taniar: Replica synchronisation in grid databases, *International Journal of Web and Grid Services*, 1(1): 87-112, 2005

L. Jourdan, M. Basseur and E-G. Talbi. Hybridizing Exact Method and Metaheuristics: A Taxonomy. *European Journal of Operational Research*, EJOR, 199(3): 620-629, 2009.

H.C. Lau, W.C. Wan, M.K. Lim, and S. Halim. A Development Framework for Rapid Meta-Heuristics Hybridization. In *Proc. of the 28th Annual International Computer Software and Applications Conference*, 362-367, 2004.

V. Reinhard and J. Tomasik. 2008. A centralised control mechanism for network resource allocation in grid applications. Int. J. Web Grid Serv. 4, 4, 461-475.

G. Ritchie and J. Levine. A fast, effective local search for scheduling independent jobs in heterogeneous computing environments. TechRep, Centre for Intelligent Systems, University of Edinburgh, 2003.

A. Santos, F. Almeida, V. Blanco, D. Diez, J. Regueira, and E. Sicilia. Towards automatic service generation and scheduling in the OpenCF project. Int. J. Web Grid Serv. 4, 4, 367 - 378, 2008

E. Talbi. A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics* 8(5): 541-564, 2002.

K. Tanaka, M. Uehara, and H. Mori. The performance evaluation of a grid using Windows PCs. Int. J. Web Grid Serv. 4, 4 (January 2008), 395-417, 2008.

D. Taniar, S. Goel. Concurrency control issues in Grid databases, *Future Generation Computer Systems*, 23(1): 154-162, 2007

D. Taniar, C. H. C. Leung, W. Rahayu, S. Goel.*High Performance Parallel Database Processing and Grid Databases*, John Wiley Sons 2008

F. Xhafa. A Hybrid Evolutionary Heuristic for Job Scheduling in Computational Grids. Springer Verlag Series: Studies in Computational Intelligence , Vol. 75, Chapter 10, 2007.

F. Xhafa, J. Carretero, A. Abraham. Genetic Algorithm Based Schedulers for Grid Computing Systems. *International Journal of Innovative Computing, Information and Control*, 3(5): 1053-1071, 2007.

F. Xhafa, J. Carretero, B. Dorronsoro and E. Alba. Tabu Search Algorithm for Scheduling Independent Jobs in Computational Grids. *Computers and Informatics*, 28(2): 237-249, 2009.

F. Xhafa, J. Carretero, L. Barolli and A. Durresi. Immediate Mode Scheduling in Grid Systems. *International Journal of Web and Grid Services*, 3(2): 219-236, 2007.

F. Xhafa, J. Carretero, L. Barolli and A. Durresi, A. Requirements for an Event-Based Simulation Package for Grid Systems. *Journal of Interconnection Networks*, 8(2): 163-178, World Scientific Pub., 2007.

F. Xhafa, L. Barolli and A. Durresi. Batch Mode Schedulers for Grid Systems. *International Journal of Web and Grid Services*, 3(1): 19-37, 2007.

F. Xhafa, J.A. Gonzalez, K.P. Dahal, A. Abraham: A GA(TS) Hybrid Algorithm for Scheduling in Computational Grids. *Hybrid Artificial Intelligence Systems Lecture Notes in Computer Science*, 2009, Vol. 5572/2009, 285-292, Springer.

F. Xhafa, J. Kolodziej, L. Barolli, A. Fundo. A GA+TS Hybrid Algorithm for Independent Batch Scheduling in Computational Grids. In *Proceedings of The 14-th International Conference on Network-Based Information Systems (NBiS-2011)*, Polytechnic University of Tirana, Albania September 7 - 9, 2011, IEEE CPS 229-235.