Pursuit of Better Results for the Examination Timetabling Problem Using Grid Resources

Christos Gogos, George Goulas, Panayiotis Alefragis and Efthymios Housos

Abstract—Examination timetabling for universities is a difficult optimization problem with its main objective being to produce the best possible schedule for every student participating. Metaheuristics are a common way of coping with this problem mainly due to their flexibility in adapting to complex real world situations. In this contribution, Grid resources are exploited in order to locate solutions that might have been unreachable in a reasonable time using the processing power of a single computer.

I. INTRODUCTION

The examination timetabling problem (ETP) is considered to be a very important hurdle for universities. Sufficient time between consecutive exams ought to be provided for every student while numerous constraints and regulations must be respected. During the last years university curriculums became more and more elaborate and complex. This can be attributed to the fact that personalized curriculums can be created as an increasing number of course choices are allowed for each student. As a result construction of high quality examination timetables (ETs) became even harder.

Several successful attempts have been tried in order to automate the creation of ETs. A recent survey about ETP can be found in [1]. Often an association is drawn with the problem of graph coloring [2]. A major event that pushed research on the ETP further was the public release of specific datasets by Carter et al. in [3]. Thereafter, numerous researchers used those datasets in order to assess their approaches in a “pure” form of the problem. Furthermore, research interest for the ETP was periodically stimulated by the PATAT series of conferences ([http://www.asap.cs.nott.ac.uk/patat/patat-index.shtml](http://www.asap.cs.nott.ac.uk/patat/patat-index.shtml)) alongside with the two International Timetabling Competitions of 2002 ([http://www.idsia.ch/Files/ttcomp2002/](http://www.idsia.ch/Files/ttcomp2002/)) and 2007 ([http://www.cs.qub.ac.uk/itc2007/](http://www.cs.qub.ac.uk/itc2007/)).

The Grid computing paradigm [4] provides to the scientific community a wealth of processing and storing resources. In this contribution the role of the Grid is investigated as a medium that facilitates the pursuit of better results for the ETP. In the process a number of different solution approaches are examined.

II. PROBLEM DESCRIPTION

The formulation of the ETP that is examined in this contribution is the one presented in the first track of the second International Timetabling Competition (ITC07). The interested reader can find details of the problem model in [5] so only a brief description of it is given here.

The ETP requires that a set of examinations be assigned to a specific combination of period and room out of a given set of available time periods and examination rooms. The solution is considered acceptable if a set of feasibility and quality constraints aiming to the maximization of the overall productivity and usability of the schedule are satisfied. While there exist formulations that the number of used time periods is a decision variable, for the problems considered in our experiments the number of time periods was fixed, spanning from some days to a few weeks. Each student has enrolled in a set of exams that for the considered problems was obligatory to be scheduled in different periods. In addition, examinations assignments had to respect constraints about the capacity of individual rooms, the compatibility of period and exam length, as well as sequencing and coexisting assignment constraints between examinations. These constraints had to be satisfied by every feasible solution. Moreover, there exist penalty generators considering the exam schedule of each individual student and the structure of the sought solution that are used to qualify the solutions. Student exam schedule related penalties exist when the solution has for the student two immediately consecutive exams, two exams in the same day or the distance among every two examinations for each particular student is less than a given threshold. The solution structure related penalties includes penalties that are imposed when examinations of mixed durations exist within individual periods, large examinations appearing later than a threshold of the available periods and penalized periods and rooms are used in the solution. For each penalty category, a user can define a penalty weight thus creating a vector called the Institutional Model Index (IMI). The objective function is the weighted sum of these penalties, which may produce low quality solutions for specific students but it appears to be a reasonable good method for comparing solutions.
III. SEQUENTIAL SOLUTION APPROACHES

During our involvement with the ETP we proposed a multi staged solution in [6] that was subsequently improved in [7]. Two major phases existed, one that constructs a feasible solution and a second that improves it. Construction occurs in a greedy randomized manner positioning firstly exams that are expected to be troublesome if scheduled at a later time. Throughout the construction of the timetable the situation of having no free combination of period and room for placing an exam is likely to occur specially for problem instances having higher values of conflict density. This is resolved by a backtracking mechanism that records exam removals that each exam is responsible for and subsequently exploits this information. When an exam cannot be scheduled without removing already scheduled exams the decision about the set of exams that should be removed is taken by examining removals that each candidate exam for elimination from the timetable has caused in the past. Construction is repeated a few times and the best solution is passed to the next phase of improvement. Local search, simulated annealing [8], shaking [9] and multiple restarts lower the value of the cost function while a stage of searching for better arrangement of exams by moving them in different rooms of the same period is activated periodically. This latter stage is implemented using Integer Programming and can only be applied to problem instances having more than one room.

The above approach produced very good results that are comparable with the results of the winner of the ITC07 competition Tomas Muller [10]. Table I presents the best cost achieved either by [7] or [10]. It should be mentioned that these results were produced under the execution time limit imposed by ITC07 (less than 10 minutes in a typical PC 2Ghz, 2Gb Ram) after 100 runs. Further information about the algorithmic approach and the results can be found at (http://www.csl.ee.upatras.gr/ett).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset 1</td>
<td>4,356</td>
<td>4,775</td>
</tr>
<tr>
<td>Dataset 2</td>
<td>390</td>
<td>385</td>
</tr>
<tr>
<td>Dataset 3</td>
<td>9,568</td>
<td>8,996</td>
</tr>
<tr>
<td>Dataset 4</td>
<td>16,591</td>
<td>16,204</td>
</tr>
<tr>
<td>Dataset 5</td>
<td>2,941</td>
<td>2,929</td>
</tr>
<tr>
<td>Dataset 6</td>
<td>25,775</td>
<td>25,740</td>
</tr>
<tr>
<td>Dataset 7</td>
<td>4,088</td>
<td>4,087</td>
</tr>
<tr>
<td>Dataset 8</td>
<td>7,565</td>
<td>7,777</td>
</tr>
</tbody>
</table>

Best results obtained after 100 runs for each problem instance

It must be noted that combination between several metaheuristics and other methods is commonly used in order to confront difficult combinatorial optimization problems. Recently hyper-heuristics [11], [12] have been proposed as a way of orchestrating the solution process by activating series of certain metaheuristics that best fit each problem. Hyper-heuristics goals can be summarized in the following: good quality of results, robustness and no need for method-specific parameter tuning. Our approach for the ETP problem shared the same goals too. It can be categorized as an explicit cooperation scheme using direct communication of autonomous programs based on the concept of a common memory [13].

IV. GRID COMPUTING

A. The EGEE Project

Grid computing, as described in [14], is “the coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations”. It is a promising computing environment to acquire significant amounts of computation resources, often needed in order to solve many instances of demanding scientific problems. There exist several coordinated efforts to create a production level Grid infrastructure, while various local-scope (i.e. country-scope) initiatives have been also created. In Greece, the HellasGrid Task Force (http://www.hellasgrid.gr) has focused on the formation of a nation-wide strategy and coordination of activities in the Grid era, in order to provide a national computing infrastructure. EGEE (http://www.eu-eggee.org) is an infrastructure project aiming to build on recent advances of Grid technology and develop a production service Grid infrastructure constantly available to scientists. EGEE has managed to attract a very large number of computational resources, Grid experts and institutions, while it has managed to overpass the European borders and attract collaboration partners from around the globe. HellasGrid computational resources are available to the EGEE infrastructure, while the Grid software is common for these infrastructures. EGEE, HellasGrid and other coordinated efforts aim to provide a standard Grid platform for e-science. Compliance with these platforms adds significant advantages, such as the possibility to reach large numbers of processors and storage area, while enjoying production level reliability and availability.

B. SchedScripter

SchedScripter provides a web services based workflow framework for distributed resource scheduling applications, over a Grid environment. SchedScripter has evolved from SchedSP and later SchedSP-WS, a framework to assist for the creation of scheduling applications as internet services [15], [16]. The workflow language supported is WS-BPEL 2.0. The main entities in SchedScripter are the Scheduling Application, the Registry and the Worker Nodes (fig. 1). The Scheduling application coordinates the solution process and defines the workflow to be executed. The Worker Nodes register themselves to the Registry, which probes them at regular intervals to verify that they are still active. The
The scheduling application finds computational resources using the registry and assigns workload to them.

Fig. 1 SchedScripter-based application overview

The services offered by each Worker Node are a File Management service, a Job Management service for running programs on the node, a JobHost service to offer a java class from a jar (java archive) as a web service and a Worker Node Management service which enables shutting down the worker node and publishing user defined services. User defined web services can be either packaged with the worker node bundle that is transferred to the remote computer or transferred and enabled while the worker node is active. For non WS-BPEL 2.0 processes, a client workload management service has been implemented, to share computational workload across the virtual cluster. The SchedScripter components are packaged in two deployment bundles, Master Node (fig. 2) and Worker Node (fig. 3).

V. DISTRIBUTED SOLUTION APPROACHES

In the distributed approaches that used the Grid as the execution environment we considered various scenarios and implemented two of them. More approaches are under development and are briefly discussed in section VII. The nature of the Grid does not guarantee the successful execution of every computational node and thus our approaches are different than the one we would have used if the solver would be used in a cluster. A study about an infrastructure that tries to alleviate uncertainties about timely execution of code components in a Grid environment can be found in [17]. In our case we deliberately denoted the importance of successful execution of each worker node. The approach is expected to produce good results even when some worker nodes fail to return results.

A. Independent Solvers with Parameter Sweep

Our first attempt, called Independent Solver with Parameter Sweep (ISPS), validated the feasibility of using the Grid as an execution environment for the ETP. Independent solvers initialized with different parameters were created in order to guide the local search procedure in different areas of the solution space. The parameters used were the random number generator seed, the initial temperature value and the temperature degradation rate used by the simulated annealing stage. Each worker node used a different initialization vector of the above parameters. After the initialization phase, each worker node used the sequential improvement phase described in the previous section in order to find and report to the manager node the best solution found in the allowed time. The manager node stored the best solution found by the worker nodes as the global solution.

B. Synchronous Continuous Improvement Approach

Our second attempt, the Synchronous Continuous Improvement Approach (SCIA), was to use a simple swarm inspired logic to coordinate the worker nodes in the solution space areas that seem to have better potential in finding a better global solution. Like the previous attempt, each worker node starts with different initial parameters but with a portion of the available solution time. Each worker node reports the solution found so far to the manager node and
waits for a new solution to continue working on improving. The manager node holds a queue of a number of solutions found so far and uses simple tournament logic between the best found solutions to send to the worker node. We do not use the single best solution found as the solution sent to each worker node because we want to diversify the searched solution space, as there is no guaranty that the best solution found in an intermediate phase will yield the best global solution. Still, the lowest cost solutions have better potential so they are the ones used by the most worker nodes.

The manager synchronizes with all the workers at the end of each stage and receives the generated intermediate solutions. There is the possibility that some workers failed in the given time window and no solution is retrieved. In this case the manager considers the worker as a non-improving worker. The process finishes when a given number of stages is completed.

VI. EXPERIMENTS

A. Execution Environment Description

The experiments were executed on the EGEE computing infrastructure, using the SchedScripter for the management of the whole process. The workflow is implemented in Java, using the workload management client API.

For the ISPS approach, the workflow is a parameter sweep. The solver and the input datasets are initially transferred to the worker node using the File Management web service. Each time a worker node is reported to be available, the solver is executed using the Job Management web service. The workflow monitors the state of the job. At the end of the job, the generated solution is collected and archived and a new problem is assigned to the worker node.

The workflow used by SCIA can also be used for the implementation of other approaches like Scatter Search. The model is consisted from a Coordinator process initiating a Problem Creator that guides the process either for a Tournament or a Scatter Search process (fig. 4). The Coordinator requests new problems from the Problem Creator, assigns them to worker nodes and reports back the results. This process is repeated until the Problem Creator decides that the process should end. At that time, the Problem Creator returns the best solution found.

For both SCIA and ISPS, the coordinator process as well as the SchedScripter master bundle were ran on a desktop Linux PC of our control, while a number of SchedScripter workers were submitted as jobs on the EGEE Grid. It is necessary to keep the execution of the SchedScripter master in a user owned PC, so that the worker nodes can have its network address when they become available in order to communicate and register with the Registry.

B. Results

For the computational experiments of the ISPS we used up to 120 worker nodes that were simultaneously solving problem instances with parameters ranging from 10 to 110 for the temperature with step at 10, 0.90 to 0.99 with step 0.01 for the temperature degradation used by the simulated annealing algorithm. For each of these runs we used different seeds for the initialization of the random number generator used internally by the solution algorithm. This is because we wanted to be able to replicate the same results despite the stochastic nature of the solution process.

![Fig. 4 Execution Environment](image)

For the SCIA approach a pool of 30 cooperating worker nodes were used. Each worker node initially generates a complete solution by running the multi-staged approach described in section III. Solutions are gathered by the coordinator process and a stochastic tournament selects one solution for each node in order to start the next stage of improvement. The best solution among the stored solutions is selected with 50% probability otherwise a weighted random selection among all stored solutions occurs. Each node is provided with different values regarding simulated annealing specific parameters. Initial temperature and cooling factor assume values for each run similar to the ones used in ISPS so as various combinations of parameters to be tested.

We managed to execute a few runs using SCIA and the results are significantly better than the results obtained by ISPS (Table II) using roughly the same processing power. Figure 5 presents the cost of the best solution returned by the problem solvers per iteration of the SCIA approach during the execution in time, for dataset 3. This example had 15 stages of improvement and lasted about 1200 seconds. Other datasets present a similar behavior during the solution process.

In comparing results from Table I and Table II it must be noted that the allowed execution time for the sequential approaches of Table I is limited while no practical limitation exists for the distributed approaches. Therefore better results were expected for ISPS and SCIA. Nevertheless results in
Table II are in general better than the results that the sequential approach of [7] could possibly locate even by allowing this approach to run for an arbitrary long period of time.

### TABLE II

**BEST RESULTS (GRID ENABLED APPROACHES)**

<table>
<thead>
<tr>
<th>Problem</th>
<th>ISPS</th>
<th>Continues Improvement Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SCIA</td>
</tr>
<tr>
<td>Dataset 1</td>
<td>4,699</td>
<td>4,699</td>
</tr>
<tr>
<td>Dataset 2</td>
<td>385</td>
<td>385</td>
</tr>
<tr>
<td>Dataset 3</td>
<td>8,546</td>
<td>8,500</td>
</tr>
<tr>
<td>Dataset 4</td>
<td>15,884</td>
<td>14,879</td>
</tr>
<tr>
<td>Dataset 5</td>
<td>2,823</td>
<td>2,795</td>
</tr>
<tr>
<td>Dataset 6</td>
<td>25,565</td>
<td>25,410</td>
</tr>
<tr>
<td>Dataset 7</td>
<td>3,975</td>
<td>3,884</td>
</tr>
<tr>
<td>Dataset 8</td>
<td>7,493</td>
<td>7,440</td>
</tr>
</tbody>
</table>

Best results obtained from different algorithmic approaches

**Fig. 5 SCIA results for dataset 3**

### VII. FUTURE DIRECTIONS

Results obtained from the experiments show clearly that there is a great potential in using GRID resources in order to locate better solutions for hard optimization problems. Currently we test an extension of SCIA in which the manager node will receive asynchronously new solutions from the worker nodes in arbitrary time points. The manager node will calculate the next solution point that each worker should use as a starting point in order to continue the optimization process. The overall process will finish when the given time expires and the last worker node will report the best solution encountered.

A particular metaheuristic that attracted our attention is Scatter Search (SS) [18] and we also plan to enhance our approach by adapting this method for the ETP as shown in Figure 4. SS is a well known metaheuristic that was originally proposed by Glover in 1965 for Integer Programming problems as a way of combining critical constraints in order to produce new “surrogate” constraints. It was also included as a stage in the original Tabu Search metaheuristic but it was seldom used in various implementations of it. At the present day, SS is considered an independent self contained metaheuristic. SS is an evolutionary algorithm maintaining a population of solutions that are recombined in order for better solutions to be achieved. Of course similarities exist with the widely known Genetic Algorithm metaheuristic but SS is less dependent on randomization, improves the solution after each recombination of solutions and systematically injects diversity into the population. A key point of SS is the concept of reference points (RPs). RP is a good solution that has been obtained from a previous solution effort. RPs are systematically combined in order for new solutions to be generated. Evolutionary algorithms usually can be well adapted for use in parallel and distributed systems. This seems to be the case for SS too. Creation of the initial population and processing of solution combinations can be greatly accelerated using distributed workers. Evidence about the applicability of SS in a parallel environment can be found in [19].

### VIII. CONCLUSION

The use of a general purpose and service oriented Grid like the EGEE for the solution of combinatorial optimization problems is an active research topic. It is clear that new algorithms are needed in order to best utilize the structure and the resources of the Grid. The results presented in this paper are evidence that this class of problems can be solved using the Grid if indeed robust algorithms that do not require frequent synchronization steps are utilized. Asynchronous and random algorithms seem as viable candidates for these problems. The Scatter Search approach that utilizes an intelligent strategy for the asynchronous coordination of the intermediate solutions is presently under research and development and will be presented in the next months.

### REFERENCES


International Timetabling Competition, Submitted to INFORMS Journal of Computing.


