Distributed Scatter Search for the Examination Timetabling Problem

Christos Gogos[1,2], George Goulas[1], Panayiotis Alefragis[1,3], Vasilios Kolonias[1] and Efthymios Housos[1]

[1]*University Of Patras. Dept. of Electrical and Computer Engineering, Rio Patras, Greece.*

[2]*Technological Educational Institute of Epirus. Dept. of Finance and Auditing, Psathaki, Preveza, Greece.*

[3]*Technological Educational Institute of Mesolonghi. Dept. of Telecommunication Systems and Networks, Varia, Nafpaktos, Greece.*

Abstract: Examination Timetabling for Universities is a problem with significant practical importance. It belongs to the general class of educational timetabling problems and has been exposed to numerous approaches for solving it. We propose a parallel/distributed solution which is based on the metaheuristic method Scatter Search combined with Path Relinking in an attempt to diversify the search procedure by producing promising new timetables. Our approach improves on the best publicly available results for the datasets of ITC2007 (International Timetabling Competition 2007-2008). The constraint of limited execution time that was imposed by ITC2007 was disregarded in an effort to pursue the best values our approach could reach. We consider this specific examination timetabling problem as a "test bed" for timetabling problems in general and we expect to provide insight for developing effective solution processes for other practical scheduling problems.

Keywords: scatter search, path relinking, examination timetabling

1. Introduction

The advent of multicore processors, cloud computing and programmable Graphical Processing Units, to name just three of recent massive processing technologies, offer nowadays abundant processing power. Difficult practical problems can be revisited and new solution methods can be sought under the presence of distributed or parallel environments of execution. Under conditions of vast availability of computational resources, mixed integer and dynamic programming approaches, which generate provable optimal solutions, become interesting. Equally interesting are metaheuristic approaches that sacrifice optimality but are in general simpler to implement and can give very good results.

The Examination Timetabling Problem (ETP) has received ample attention and numerous approaches from and across various disciplines have been proposed for solving it. Some of the approaches that have given satisfactory results are: Constraint Programming (David, 1998), Hybrid methods (Qu and Burke, 2009), Hyper Heuristics (Pillay and Banzhaf, 2008) and various

Metaheuristics (Ersoy et al., 2007). In this contribution the ITC2007[1] model of the Examination Timetabling Problem is solved by a Scatter Search metaheuristic and the whole process is undertaken by our distributed execution framework called SchedScripter. The solutions that we have obtained indicate that significant potential of using analogous methods to similar problems exists. We also observed that the benefit of reaching better solutions was complemented by the robustness of the procedure.

The rest of the paper is organized as follows. Section 2 describes the ETP. Section 3 presents an introduction to parallel/distributed execution environments and metaheuristics. The SchedScripter framework is also presented in the same section. SchedScripter is used for communications and workload management needed by the distributed Scatter Search application. Section 4 describes the distributed Scatter Search solution approach, with subsections describing Scatter Search and Path Relinking. Section 5 presents the experimental results. Section 6 concludes the paper asserting that distributed Scatter Search can outperform various single processor approaches in terms of solution quality.

2. Problem Description

The ETP belongs to the general class of timetabling problems which are known to be NP-complete under certain conditions (Schaerf, 1999). Several different formulations of ETP exist ranging from rather simple ones to more complicated (Qu et al., 2009). Historically, early formulations considered only the avoidance of exam conflicts thus drawing parallelism between ETP and graph coloring problems (Carter, 1996) while gradually details regarding rooms, periods and exams were added. The main objective of the ETP is to produce timetables giving adequate time between exams for all participating students.

In our contribution, we use the ITC2007 formulation of the problem. Under this formulation a set of exams has to be scheduled in a set of time periods and rooms while at the same time a number of hard constraints have to be satisfied. Hard constraints are avoidance of conflicts between exams for all students, respect of room capacities, matching of exam duration with assigned period duration and various constraints regarding ordering between exams and pre-assignment of exams to certain rooms. The quality of the solution is measured against cost penalties imposed by a set of soft constraints. Examples of soft constraints considered are the presence of two exams in a row and two exams in the same day for a particular student, early scheduling of exams with large audiences, mixing of exams with different durations in the same room and the usage of rooms and periods that have been marked as undesirable. Period spread is also a soft constraint meaning that occurrences of consecutive exams for each student within a predefined range of periods are penalized. Every soft constraint is related with a weight that prescribes its penalty. Each university defines the weight of every constraint according to its preferences thus creating a vector called the Institutional Model Index (IMI). The objective function is a weighted sum of the soft constraint violations according to the weights defined in IMI.

[1] http://www.cs.qub.ac.uk/itc2007

A thorough description of the ITC2007 ETP formulation can be consulted in (McCollum et al. 2009a) while further details regarding ITC2007 issues in general can be found in (McCollum et al. 2009b). Twelve datasets were provided for benchmarking and data associated with them are presented in Table 1. The last four of the datasets were characterized as hidden indicating the intention of the organizers to make them publicly available after the completion of the competition. The 12 datasets present significant variation in their characteristics which is also observed in real life examination timetabling problems (McCollum, 2007). Therefore, a generic successful algorithmic approach should not make any assumptions about specific values.

ITC2007-DATASETS														
	Exams	Students	Periods	Rooms	Period HC	Room HC	Two In A Row Penalty	Two In A Day Penalty	Period Spread Penalty	No Mixed Durations Penalty	Number Of Largest Exams	Number Of Last Periods To Avoid	Frontload Penalty	Conflict Density
Dataset 1	607	7891	54	7	12	0	7	5	5	10	100	30	5	5.05%
Dataset 2	870	12743	40	49	12	2	15	5	1	25	250	30	5	1.17%
Dataset 3	934	16439	36	48	170	15	15	10	4	20	200	20	10	2.62%
Dataset 4	273	5045	21	1	40	0	9	5	2	10	50	10	5	15.00%
Dataset 5	1018	9253	42	3	27	0	40	15	5	0	250	30	10	0.87%
Dataset 6	242	7909	16	8	23	0	20	5	20	25	25	30	15	6.16%
Dataset 7	1096	14676	80	15	28	0	25	5	10	15	250	30	10	1.93%
Dataset 8	598	7718	80	8	20	1	150	0	15	25	250	30	5	4.55%
Dataset 9	169	655	25	3	10	0	25	10	5	25	100	10	5	7.84%
Dataset 10	214	1577	32	48	58	0	50	0	20	25	100	10	5	4.97%
Dataset 11	934	16349	26	40	170	15	10	50	4	35	400	20	10	2.62%
Dataset 12	78	1653	12	50	9	7	35	10	5	5	25	5	10	18.45%

Table 1. Datasets characteristics

3. Parallel / Distributed Execution Environments and Metaheuristics

It is often the case that real life problems generate problem instances that require vast amounts of CPU time to create optimal feasible solutions. Although the use of metaheuristics allows significant reduction of the search process computational complexity, the wall clock time is still the major performance measurement. End-users in many application areas require that high quality solutions be obtained as soon as possible. In such applications the cost of hardware resources that may be required is considered a minor issue. A detailed presentation of the possibilities in metaheuristic design can be found in (Talbi 2009). Recently, parallel processing has again gained significant attention as various technology advancements has been performed in processor design (multicore processors, GPU computing) and interconnecting networks (e.g. Infiniband). Moreover, Grid technologies allow the exploitation of vast amounts of usually volatile loosely coupled computational resources, creating an opportunity to exploit such architectures for the design and implementation of parallel metaheuristics.

Many aspects, design decisions and goals have to be considered during parallel metaheuristic algorithm design. The most usual goal is to speed up the search process or the improvement of the quality of the obtained solutions. The former allows the design of real-time or interactive optimization methods, while the latter allows the cooperating metaheuristics to obtain better convergence characteristics while reducing search time. Researchers or practitioners using parallel processing have managed to improve the robustness of the obtained solutions for various difficult

practical problems. This is usually achieved by reducing the sensitivity of the metaheuristics to their parameters and by managing to examine the search space in greater detail. During parallel metaheuristic design, questions such as exchange decision criterion (when?), exchange topology (where?), information exchanged (what?) and integration policy (how?) have to be answered. Detailed presentation of various parallel metaheuristics can be found in (Alba, 2005).

Our approach was to create a middleware architecture that isolates parallel algorithm design components from the mapping to parallel architectures, providing a toolbox to rapidly create different parallelization approaches.

3.1 SchedScripter framework

SchedScripter (Gogos et al., 2009), (Goulas et al., 2009) is a software framework to assist in the development of distributed, grid-based, human resources scheduling applications. SchedScripter covers an area between loosely-coupled grid-based workflow systems (Yu and Buyya 2005) and message passing libraries. Grid workflow systems create execution workflows based on independent jobs and their data dependencies, while message passing libraries assist in the creation of parallel/distributed applications using tight message passing protocols like the MPI (Gropp et al., 1999). SchedScripter applications main components are the SchedScripter registry, the SchedScripter worker nodes and the application coordination process (Figure 1). SchedScripter installs a web service container on every worker node to provide a set of services designed to allow the worker node to be considered as a single process of a worker collection. A single master node provides a registry for the worker node services and ensures that workers remain available. All SchedScripter services are offered as XML Web Services for interoperability and ease of access. A SchedScripter-based application needs to access the registry to find resources and then transfer tasks and code to worker nodes. This process is assisted by an application level task scheduler offered in the SchedScripter API.

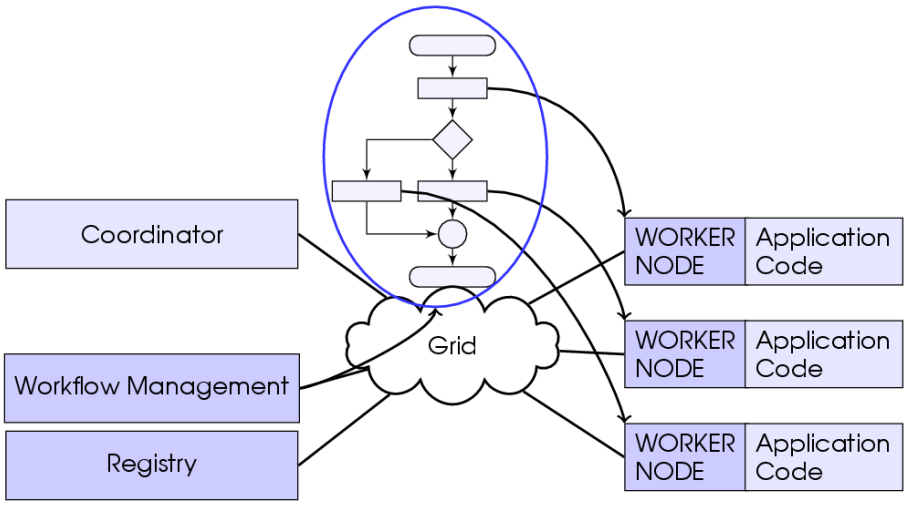

Figure 1. SchedScripter architecture

A natural coordination scheme for web services, supported by SchedScripter, is the use of Business Process Execution Language (BPEL[2]) workflows. BPEL allows for web service message interactions providing programming language constructs. BPEL tools offer graphical representation of application workflows which is a great visualization tool. In order to assist resource discovery and utilization using BPEL, the SchedScripter application-level task scheduler is available as a web service. Furthermore, since BPEL processes are themselves web services, BPEL can offer layers of abstraction, composite services and reuse. The SchedScripter master includes Apache ODE[3] as a BPEL engine, compatible with both BPEL versions, BPEL4WS 1.1 and WS-BPEL 2.0.

While web services and BPEL are an easy way to create distributed applications, it was clearly demonstrated that it is common for developers to have difficulties in understanding the concepts of Service Oriented Architecture so as to use web services and BPEL effectively. To narrow this knowledge gap, SchedScripter tries to abstract the process of applying specific distributed application patterns and offers them as Application Templates in the form of a Java API. The main patterns supported are the master/worker paradigm and the swarm pattern. The master/worker is a very popular, centralized pattern where a single master process splits the workload into tasks and assigns them to worker nodes. The swarm pattern, also usually referred as peer-to-peer, assumes a set of independent processes, each of them deciding on the task to accomplish in order for the whole swarm to solve the general problem. These processes communicate frequently with broadcasted messages, in order to publish their findings and provide hints to the swarm, which may or may not be used. The swarm, in SchedScripter, has a single master, whose role is to monitor message exchanges and store the result at the end of the process. While SchedScripter was originally developed to be used in the EGEE[4] grid infrastructure, the framework is generic and can be used outside EGEE as well, even in a cluster environment without grid middleware. Indeed, the results of this paper were collected during runs on the small cluster of servers of a newly acquired blade system running Ubuntu Karmic 9.10 server operating system.

4. Solution Process

The solution process that we have chosen is based on metaheuristic Scatter Search and uses data structures and algorithms from our previous work described in (Gogos et al, 2010). A stage of Simulated Annealing and a stage of Shaking are combined in a single improvement phase that is employed whenever an attempt to improve a solution occurs. In the Shaking stage a set of exams is formed based on the cost per student contribution of each exam giving preference to higher values. The selected exams are removed and then scheduled once again causing distortion to the timetable.

---

[2] http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

[3] http://ode.apache.org

[4] http://www.eu-egee.org

4.1 Scatter Search

Scatter Search is a well known metaheuristic that was originally proposed by Glover for Integer Programming problems as a way of combining critical constraints in order to produce new "surrogate" constraints (Glover, 1977). It was also included as a stage in the original Tabu Search metaheuristic but it was seldom used in various implementations of it. Paper (Glover 1998) played a key role in establishing Scatter Search as an independent self contained metaheuristic. Detailed descriptions of Scatter Search can be found in (Glover et al., 2002), (Glover et al., 2003) and (Laguna and Marti, 2006) while in (Laguna and Armentano, 2000) several practical advices about implementing Scatter Search are illustrated. It has to be mentioned that Scatter Search has been recently employed in (Mansour et al., 2009) for tackling a modified formulation ETP giving promising results.

Scatter Search is classified as an evolutionary algorithm. It maintains a population of solutions that are recombined in order for better solutions to be achieved. Similarities exist with the widely known Genetic Algorithm metaheuristic but Scatter Search maintain smaller population of solutions, is less dependent on randomization, improves the solution after each recombination of solutions and systematically injects diversity into the population. A key point of Scatter Search 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. Scatter Search starts by generating a pool of good and diverse solutions. From this pool a reference set (RefSet) is formed that contains not only the best solutions but also solutions that are different from the other already included in the RefSet. Next, new solutions are formed by combining members of the RefSet. Each produced solution is improved through heuristic methods. The best improved solutions are inserted into the RefSet causing the removal of inferior solutions. The procedure continues in an iterative manner creating combinations of the newly inserted solutions with already existing solutions of the RefSet. When no new solutions can be inserted into the RefSet then either the procedure stops or certain actions are initiated that diversify RefSet by replacing solutions with less good but significantly different ones. The general template of Scatter Search consists of five methods forming a cycle of continuous improvement over a set of solutions. Our implementation of the Scatter Search is shown in (Figure 2) where each solution is depicted by a hexagon. The role that each method plays in solving the ETP is presented in the following paragraphs.

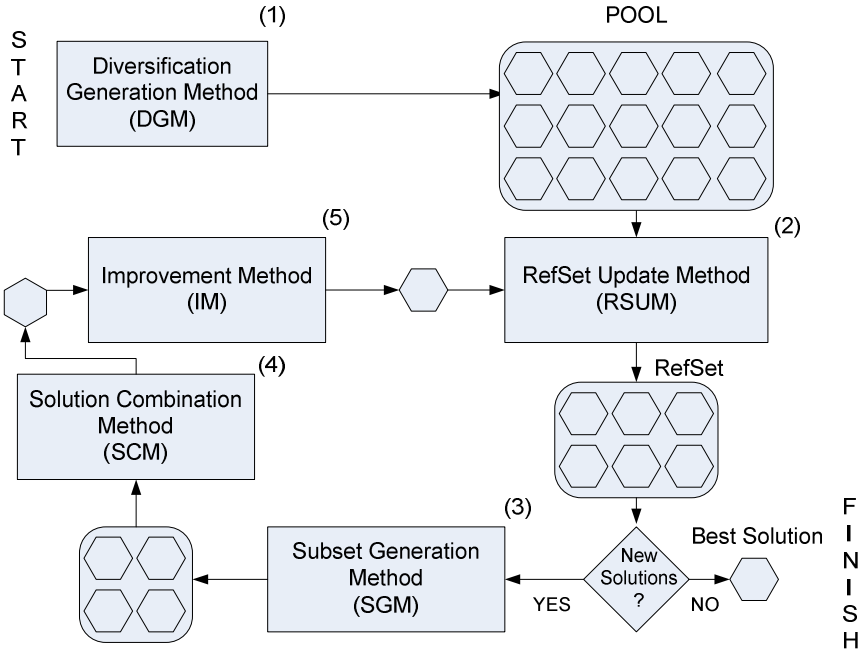

Figure 2. Scatter Search 5 method template

Diversification Generation Method (DGM).

The purpose of this method is to generate a collection of diverse timetables. In our approach we simply loaded a pool with 100 solutions from past experiments done in (Gogos et al., 2010). Alternatively, the pool could be generated on demand by running several time bounded constructions and improvements while modifying various parameters like construction over improvement time, simulated annealing cooling schema etc. Experiments showed that the exact method of populating the pool does not hinder the robustness of the approach. Then, an initial RefSet is constructed by selecting exams from the pool. Half of RefSet solutions are selected from the pool based on their cost. The other half is completed by selecting solutions that exhibit the biggest dissimilarity with the timetable of the RefSet that is more similar to the selected solution. It has to be noted that diversification is not the same as randomization because the whole process is biased towards selecting solutions that differ from other ones.

Improvement Method (IM).

Improvement of each solution is undertaken by a cycle of two stages. These stages are "simulated annealing" and "shaking". Simulated annealing allows, in a systematic way, moves to inferior solutions thus enabling the possibility of escape from local best values basins. On the other hand the purpose of the shaking stage is to dislocate the current solution so as new searches for better values to start from yet unexplored points of the search space. In our distributed execution approach a master/slave schema is employed. The improvement method executes in parallel by worker nodes while a coordinator is responsible for collecting results and producing new jobs under the SchedScipter framework described in Section 3.

RefSet Update Method (RSUM).

Solutions become members of the RefSet based on their cost. Inferior timetables may also be accepted provided that they are sufficiently different from existing timetables in RefSet. The

similarity of two solutions is computed based on the number of exams that are scheduled in the same period and room in both solutions.

Subset Generation Method (SGM).

This method operates on the RefSet and produces a subset that will be used in order for solutions to be combined. In our approach pairs of solutions are formed ensuring that each pair has not already been examined in the past.

Solution Combination Method (SCM).

Path Relinking occurs in this method. Pairs of timetables are combined in order to construct new solutions. A given timetable is gradually transformed to another timetable (guiding solution) by repositioning exams to periods and rooms. In order for the solution to be in the feasible area backtracking moves that remove offending exams and then reschedule them might be necessary.

4.2 Path Relinking

Suppose the existence of two complete timetables A and B assuming that A is superior to B considering cost value. The objective during Path Relinking is to gradually transform A to B hoping that promising timetables will arise during the process (backward relinking). At each step of the procedure, certain attributes of A are modified so as to become identical with those of B. During the transformation process intermediate, but complete, solutions are stored whenever they are considered to be of value for subsequent stages of the overall process.

More specifically timetable A becomes timetable B gradually in cycles. In each cycle a group of exams are selected from timetable A and rescheduled to new periods and rooms so as to comply with those found in timetable B. It is possible that the rescheduling of a group of exams in A will result in cascaded repositions of other exams in order to retain the feasibility of the solution. Whenever the resulting timetable has more differences than the original, a rollback is performed to the previous state of timetable A and an extra exam is added to the initial group of exams. The use of the Command design pattern (Gamma et al., 1994) made rollback moves easy to be programmed. The pseudo-code of the above process follows:

Step 1: Set N=1. Find differences between A and B and put them on set DIFFS. Set MIN_SIZE=|DIFFS|. Set LEVEL=MIN_SIZE

Step 2: If MIN_SIZE=0 then terminate. Else all exams of A which are members of DIFFS that can be scheduled to the same period and room as in B are repositioned accordingly.

Step 3: N exams are selected randomly from set DIFFS. For each selected exam other exams must be removed from the timetable so as to allow the scheduling of the selected exam to the period and room dictated by B. Priority for removal is given to exams that are not scheduled in the same period and room as in B. If removing these exams is not sufficient extra exams are selected on the basis of evoking minimum removals of other exams in previous stages of the algorithm. Extra exams are selected one by one until the selected exam can be scheduled in the correct period and room. Each one of the N exams are scheduled in the period and room found in solution B, while a set of exams called UNSCHED contains exams that no longer belong to the timetable.

Step 4: All exams of set UNSCHED are scheduled to any of the available periods and rooms until set UNSCHED become empty. In the process, removal of already scheduled exams is possible. If this is the case then exams for removal are selected based again as in Step 3 on the history of removals caused by each candidate exam. It should be mentioned that during this step no restrictions apply on which exams could be removed.

Step 5: Update set DIFFS by finding differences between solution A and B. If |DIFFS|>=MIN_SIZE then N=N+1 else MIN_SIZE=|DIFFS|.

Step 6: If MIN_SIZE<LEVEL then LEVEL=MIN_SIZE and N=1. Return to Step 2.

5. Experiments

While the application development and debugging has been carried out on a small cluster of desktop PCs and the South East Europe part of the EGEE infrastructure, the results were obtained running on a brand-new IBM Blade system. This system consists of 14 independent servers in a single enclosure, each of them powered by a quad core hyper-threaded CPU Xeon processor based on Nehalem architecture. These processors were coordinated by another server, which we call the head node, a dual CPU quad core Xeon based on Clovertown architecture.

The head node hosts the SchedScripter master, which includes the services registry, as well as the Distributed Scatter Search application coordinator. The 14 blade servers host the worker node process and in order to fully exploit the 8 virtual cores (4 real ones hyper-thread), 8 processes start on each server. This system was installed recently, so there was little time to measure the impact of possible memory bandwidth bottlenecks. The different coordination process runs were managed using Sun Grid Engine and a local queue with one slot on the head node. Several configuration experiments were taking place at one or two nodes while the experiments were running, removing these servers eventually from the available resources, so the resource pool was varying from 12 to 14 blades, or from 96 to 112 worker processes. The same experiment running on the EGEE grid would suffer much more uncertainty in resource numbers, as on the grid resources enter and leave at random times during runtime as the system lacks rendezvous mechanisms.

Every time the coordinator finds a new worker node, it transfers the necessary executables and instructs it to load the problem dataset, which is about a minute long process. After this initial step, specific improvement tasks are sent to the worker nodes. An improvement task is an instruction to perform local search (Simulated Annealing and Shaking), starting from a specific solution with a specific set of local search parameters and a timeout period. When the worker node finishes the local search, it sends the resulting solution back to the coordinator, using a web service.

SchedScripter, being a grid framework, is highly tolerant to resource failure events and dynamically resizes the resource pool at runtime. Distributed Scatter Search Coordinator as well, has been created as a fault-tolerant process, able to dynamically resize its workers pool. Indeed, since the biggest part of the debug sessions took place in the EGEE grid, where we submit about a couple of hundred of worker jobs, these jobs start at random times as the grid workload management system decides. These potential problems in reliability lead us to include fault-tolerance as an inherent part of the coordination approach.

Table 2 displays the best results for each dataset of the problem under the runtime limit of about 10 minutes imposed by the ITC2007 competition. The second column contains the best results collected from the winner of the competition, Tomas Muller, after 100 runs (Muller, 2008). It should be noted that no data exist in this paper for the former hidden datasets because at that time those datasets were not available. The next two columns are the best results collected after 51 independent runs as cited in (McCollum et al., 2009). The last column of the table depicts the best results over 100 runs as cited in (Gogos et al., 2010). The last 4 values of the last column are not included in (Gogos et. al., 2010) so new runs for those datasets were made and the values recorded are the best results collected over 100 runs.

Instance	(Muller, 2008)	(McCollum et al, 2009) (Post ITC2007)	Muller (Post ITC2007)	(Gogos et al, 2010)
Dataset 1	**4.356**	4.663	4.370	4.775
Dataset 2	390	405	**385**	**385**
Dataset 3	9.568	9.064	9.378	**8.996**
Dataset 4	16.591	15.663	**15.368**	16.204
Dataset 5	2.941	3.042	2.988	**2.929**
Dataset 6	25.775	25.880	26.365	**25.740**
Dataset 7	4.088	**4.037**	4.138	4.087
Dataset 8	7.565	**7.461**	7.516	7.777
Dataset 9	X	1.071	1.014	***964***
Dataset 10	X	14.374	14.555	***13.203***
Dataset 11	X	29.180	31.425	***28.704***
Dataset 12	X	5.693	5.357	***5.197***

Table 2. Best results under time limit

The configuration used in our experiment with Scatter Search included a RefSet of 20 solutions, collection of 4 solutions during each Path Relinking, 120 seconds available for each improvement attempt and total runtime of 4 hours. In Table 3 the results of our approach are presented and compared alongside with results obtained in (Gogos et al., 2009) with the current approach giving better results in all datasets. The last column of the table shows the percentage of improvement achieved over best results of Table 2. For all 12 datasets improvement was achieved, while for certain datasets improvement was beyond our initial expectations.

Instance	(Gogos et al, 2009)	Current approach (Scatter Search)	Percentage of improvement over best results of Table 2
Dataset 1	4.699	**4.128**	5,23%
Dataset 2	385	**380**	1,30%
Dataset 3	8.500	**7.769**	13,64%
Dataset 4	14.879	**13.103**	14,74%
Dataset 5	2.795	**2.513**	14,20%
Dataset 6	25.410	**25.330**	1,59%
Dataset 7	3.884	**3.537**	12,39%
Dataset 8	7.440	**7.087**	5,01%
Dataset 9	X	**913**	5,29%
Dataset 10	X	**13.053**	1,14%
Dataset 11	X	**24.369**	15,19%
Dataset 12	X	**5.095**	1,96%

Table 3. Best results under no hardware or time limits

In order to be fair in our comparisons a few points have to be stressed out. The execution environment is different between approaches included in Table 2 and Table 3. In the former case a

time limit of about 10 minutes was given while in the latter no practical time limit was imposed. Furthermore, the number of runs is different. In Table 2 the results are the best collected from 51 or 100 runs while in Table 3 the results are collected from a single experiment that consumed more processing power than all runs in each dataset of Table 2. However, we believe that our approach demonstrates an effective method for using additional time and CPU resources if they are available.

6. Conclusions

Evolutionary algorithms usually can be well adapted for execution in parallel or distributed systems. This is the case for Scatter Search too. Creation of the initial population and processing of solution combinations can be greatly accelerated using distributed workers. In this contribution we have proposed a Scatter search technique for tackling the examination timetabling problem harnessing the processing power of a very powerful computer system. Results show improvements over best known values for all datasets of the ITC2007 ETP problem, which are for some cases significant. We acknowledge the fact that results we compare with were obtained under a small fraction of the processing time that we spent but we believe that for certain problems this extra time can be allocated. Consequently, whenever "very good" solutions for difficult practical problems are of significant importance over simply "good" solutions experimentation with scatter search or other evolutionary metaheuristics combined with distributed execution environments have increased possibilities to pay off.

References

Alba, E. Editor (2005). Parallel Metaheuristics, A New Class of Algorithms, ISBN 978-0-471 - 67806-9, John Wiley & Sons, Inc.

Carter M, Laporte G, Lee S (1996). Examination Timetabling: Algorithmic Strategies and Applications. Journal of Operational Research Society, Volume 47, pp 373-383.

David P. (1998). A Constraint-Based Approach for Examination Timetabling Using Local Repair Techniques. In: E.K. Burke and M.W. Carter (eds). Practice and Theory of Automated Timetabling: Selected papers from the 2nd International conference, LNCS, Volume 1408, pp 169-186. Springer-Verlag, Berlin, Heidelberg.

Ersoy E., Ozcan E. and Sima Uyar A. (2007). Memetic Algorithms and Hyperhill-Climbers. Proceedings of the 3nd Multidisciplinary International Conference on Scheduling: Theory and Applications, MISTA07, pp 159-166.

Gamma E., Helm R., Johnson R., Vlissides J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series. ISBN-13: 978-0201633610.

Glover F (1977). Heuristics for Integer Programming Using Surrogate Constraints. Decision Sciences, Vol 8, No 1, pp. 156-166.

Glover F (1998). A template for scatter search and path relinking. In: Hao JK, Lutton E, Ronald E, Schoenauer M, Snyers D (eds). Artificial evolution. Lecture notes in computer science, vol 1363.

Glover F, Laguna M, Martí R (2000) Fundamentals of scatter search and path relinking. Control Cybern 29(3):653–684.

Glover F, Laguna M, Martí R (2002) Scatter search. In: Ghosh A. Tsutsui S (eds) Theory and applications of evolutionary computation: recent trends. Springer, Berlin, pp 519–529

Glover F, Laguna M, Martí R (2003) Scatter search and path relinking: advances and applications. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics. Kluwer Academic, Dordrecht, pp 1–36

Gogos C, Alefragis P and Housos E (2010). An Improved Multi-Staged Algorithmic Process for the Solution of the Examination Timetabling Problem. Annals of OR, DOI: 10.1007/s10479-010-0712-3.

Gogos C, Goulas G, Alefragis P and Housos E (2009). Pursuit of Better Results for the Examination Timetabling Problem Using Grid Resources, 2009 IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched), Nashville, Tennessee, USA, 30 Mar-2 Apr 2009, pp 48-53, DOI:10.1109/SCIS.2009.4927014.

Goulas G, Gogos C, Alefragis P and Housos E (2009). SchedScripter: Workflows for Grid-based Human Resources Scheduling Applications, 4th EGEE User Forum/OGF 25 and OGF Europe's 2nd, Catania, Sicily, Italy, 2-6 March 2009

Gropp W, Lusk E and Skjellum A (1999). Using MPI: Portable Parallel Programming with the Message Passing Interface, 2nd Edition, MIT Press.

Laguna M and Armentano V (2005). Metaheuristic Optimization via Memory and Evolution Tabu Search and Scatter Search. Lessons from applying and experimenting with Scatter Search, pp. 229-246. Operations Research Computer Science Interfaces Series, Vol. 30.

Laguna M. and Marti R. (2006). Metaheuristic Procedures for Training Neural Networks edited by Alba E. and Marti R. Scatter Search. Springer Science+Business Media, LLC.

Mansour N, Isahakian V and Ghalayini I. (2009). Scatter Search Technique for Exam Timetabling. App Intell, DOI 10.1007/s10489-009-0196-5.

McCollum B. (2007). A Perspective on Bridging the Gap between Theory and Practice in University Timetabling. In: PATAT 2006, LNCS 3867, pp 3-23, ISBN 978-3-540-77344-3. Berlin: Springer.

McCollum B., McMullan P., Burke E., Parkes A., Qu R. (2009a). A New Model for Automated Examination Timetabling. Submitted to post PATAT Annals of OR.

McCollum B., Schaerf A., Paechter B., McMullan P., Lewis R., Parkes A., Di Gaspero L., Qu R., Burke E. (2009b). Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition. INFORMS Journal of Computing 2009. DOI:10.1287/ijoc.1090.0320.

Muller T. (2008). ITC 2007: Solver description. Proceedings of the 7th International Conference on Practice and Theory of Automated Timetabling. University of Montreal, Canada.

Pillay N. and Banzhaf W. (2008). A Study of Heuristic Combinations for Hyper-Heuristic Systems for the Uncapacitated Examination Timetabling Problem. European Journal of Operational Research. DOI:10.1016/j.ejor.2008.07.023.

Qu R, Burke E, McCollum B Merlot L and Lee S (2009). A Survey of Search Methodologies and Automated System Development for Examination Timetabling. Journal of scheduling, 12(1), pp 55-89. DOI:10.1007/s10951-008-0077-5.

Qu R. and Burke E.K. (2009). Hybridizations Within a Graph Based Hyper-Heuristic Framework for University Timetabling Problems. Journal of Operational Research Society. DOI:10.1057/jors.2008.102

Schaerf A. (1999). A Survey of Automated Timetabling. Artificial Intelligence Review, Volume 13, pp 87-127. Kluwer Academic Publishers, Netherlands.

Talbi, El-Ghazali(2009). Metaheuristics, from Design to Implementation, ISBN 978-0-470-27858-1, John Wiley & Sons, Inc.

Yu J. and Buyya R. (2005). A Taxonomy of Workflow Management Systems for Grid Computing, Journal of Grid Computing, Volume 3, Numbers 3-4, Pages: 171-200, Springer, New York, USA.