The Interleaved Constructive Memetic Algorithm and its Application to Timetabling

Ender Özcan, Andrew J. Parkes, Alpay Alkan[a]

ASAP Research Group, School of Computer Science, University of Nottingham, NG8 1DY, UK, e-mail: exo and ajp@cs.nott.ac.uk.

[a]*Department of Computer Engineering, Yeditepe University, Istanbul, Turkey, e-mail: alpayalkan@gmail.com.*

Abstract

Timetabling problems are well known NP-hard constraint satisfaction, and real-world cases often have complicated and challenging structures. For such problems, we present a new hybrid method, "Interleaved Constructive Memetic Algorithm" (ICMA) that interleaves memetic algorithms with constructive methods. ICMA works using an active subset of all the events. Starting with a few events, in multiple construction stages ICMA increases the active set to eventually include all of them. At each stage, a memetic algorithm (MA) is applied to improve the current partial solution before the next construction step. We also describe a real-world course timetabling problem, the "Preparation School Timetabling Problem" (PSTP), which is of particular interest because it has a highly hierarchical structure arising from various organisational requirements. An important advantage of ICMA is that both the constructive heuristics and MA can be tailored to exploit such hierarchical structures. We give empirical results showing that ICMA performs better than the corresponding conventional MA on the PSTP.

Keywords: Genetic algorithms; timetabling; scheduling; metaheuristics; memetic algorithms.

1. Introduction

Timetabling problems are NP-complete and often provide significant real-world challenges [1]. Naturally, memetic algorithms (MAs) have been applied to such problems: A survey on timetabling and that particularly discusses the use of MAs can be found in [2]. As common in practical problems, timetabling problems have many different kinds of constraints, and they can be *hard* or *soft*. Hard constraints represent unacceptable situations, whereas soft constraints are less essential preferences. Hill-climbing, and so also MAs, can be designed explicitly to target the different constraint types. For example, Alkan and Özcan (2003) [3] introduced a violation directed hierarchical hill climbing method (VDHC). VDHC performs a local search over the constraint oriented neighborhoods as described in Viana, Pinho de Sousa and Matos (2003) [4] based on the constraint types, their violations and structure of the problem. Özcan (2007) [5] extended the study and suggested a heuristic template for designing a set of operators. Before this, a variety of genetic operators and self-adjusting hill-climbers implicitly respecting this template had already been investigated within MAs for solving different timetabling problems in [6], [7] and [8]. The VDHC methods within the MAs that managed multiple constraint based hill climbers turned out to be successful in timetabling.

As standard with MAs, this previous work on VDHC always used the complete problem, that is, it would be working simultaneously on finding suitable reassignments for all of the events. However, it is well-known that constructive methods can also be effective. Such methods work by finding solutions for a subset of events, and then increasing this subset; that is, they work with partial assignments instead of complete assignments. Working with partial assignments and constructive techniques has advantages. Firstly, it allows the use of powerful constructive heuristics and these can incorporate domain knowledge and so be highly effective. Secondly, we also believe that it has the potential advantage of often focussing search effort onto smaller subproblems where the computational search effort might be used more efficiently than on the full problem. This suggests that it could be useful to build a hybrid that tries to get the best of both MAs and constructive methods. Naturally, a randomised constructive method could be used to create new individuals for populations and then MA is applied to the full individual. However, here we give a method in which the MA and construction are tightly connected in that applications of MA and construction are interleaved. We call this new hybrid method the "Interleaved Constructive Memetic Algorithm" (ICMA). This paper will describe this method and show its successful application to a real problem called the "Preparation School Timetabling Problem" (PSTP) (based on an initial study which can be found in [9]).

The "Preparation School Timetabling Problem" PSTP, that we will describe, is an example of a Course Timetabling Problem (CTP). These are a subset of timetabling problems that require an arrangement of resources for the teaching of courses, such as teaching events (lectures, classes, etc) to times and rooms, as well as teachers to deliver them. (The PSTP was analyzed initially by Alpay Alkan during his studies on timetabling). Such problems have a wide variety of constraints (these can be categorized further in practical timetabling [10]) and so constraint-directed hill-climbers within an MA framework can be applied. However, the PSTP is also highly hierarchically structured as a result of the administrative and organisational system within the schools. One key advantage we will see of ICMA is that it can successfully exploit this hierarchical/clustered structure. Generally, we believe that this ability to include domain specific heuristics during its interleaved construction, could give ICMA an advantage in many domains besides timetabling.

The structure of the paper is as follows. In Section 2 we briefly survey existing relevant work in timetabling, and the basics of memetic algorithms. Section 3 presents the PSTP, and in particular its hierarchical structure and constraints. The general ICMA method, in the context of timetabling, is given in Section 4. Section 5 gives the details of how ICMA is implemented for the specific case of the PSTP. Experimental evidence for the effectiveness of ICMA is given in section 6: The performance of ICMA is compared to the conventional one both using a VDHC instance on real PSTP instances. The conclusions in section 7 discuss ideas of what underlies the success and avenues for future research.

2. Related Work

2.1. Memetic algorithms

Memetic algorithms (MAs), also referred to as hybrid genetic algorithms (GAs) represent a set of population based approaches that combine genetic algorithms [11] and local search [12]. A *meme* represents the local search component in an MA. There are many studies showing that the use of a meme can improve the performance of a GA and different memes might yield different performances [5]. The description of different types of memetic algorithms and their categorisation can be found in [13], [14]. In this paper, a different approach based on a memetic algorithm is proposed for solving PSTPs.

Figure 1 illustrates the basic steps of a generic memetic algorithm. An *individual* (chromosome) in a GA represents a candidate solution for a given problem. Each *gene* that composes an individual receives an *allele* value from a set of values. For example, in a binary representation, a gene gets an allele value from the set {0, 1}. In an iterative cycle, a randomly generated *population* of individuals evolves towards an optimal solution by passing through a set of genetic operators, namely *crossover*, *mutation* and *replacement*. During crossover, selected individuals named as *mates*, exchange their genetic materials with a given probability (p_c). Mutation randomly perturbs some alleles. The traditional mutation operator scans each offspring bit by bit and inverts the ones with a given probability (p_m). The replacement process decides which individuals will survive to the next generation. In a conventional MA, a meme (e.g., a *hill climber*) is applied after the mutation. An MA requires an evaluation function referred to as *fitness function* as a vital component. In a usual mate selection scheme, it is more likely for an individual being selected as a mate that has a better fitness value.

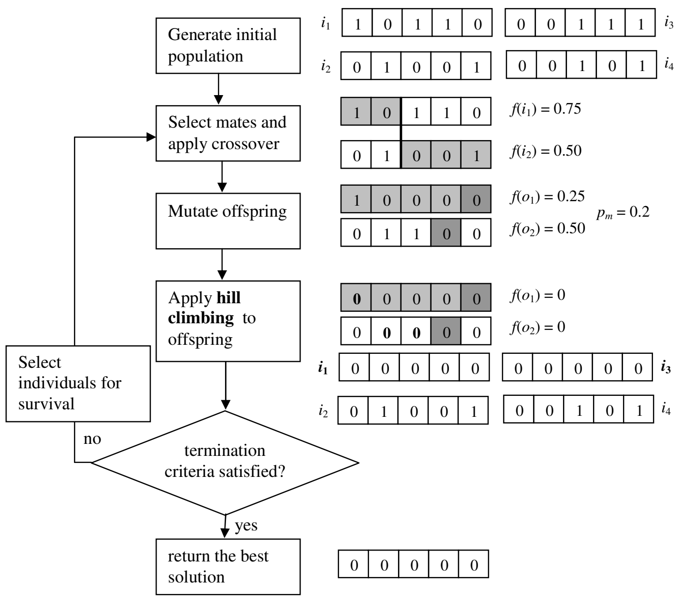


Figure 1: Flowchart of the conventional memetic algorithm and illustration of a single iteration for solving a sample problem

Figure 1 also provides a snap shot of how each component of an MA is employed in a single iteration using a toy problem of minimizing the number of 1s. The chromosome length is set as five. Four individuals are randomly generated at first. The fitness function in the example multiplies each bit by 0.25 and accumulates all results. Mate selection method chooses the first and second individuals for crossover. Then the traditional one-point crossover is applied to them. A random crossover locus is generated. At the second position, genetic materials are swapped producing two different offspring. The mutation probability is taken as 0.2. A single bit is inverted in each offspring in the example. Then, the fitness values of the offspring are computed. Hill climbing systematically searches neighborhoods for a given offspring to obtain better solutions. In the example, the same solutions are achieved.

2.2. High-school course timetabling and metaheuristic solutions

The university and high-school course timetabling problems are the subsets of CTPs that are commonly dealt with by numerous researchers. In both cases, a set of courses is to be scheduled using a discrete timetable having a fixed number of periods at each weekday. A group of students belonging to the same group; i.e., grade (department-term), has to attend the same set of lessons (classes, courses). This grouping can be considered as a curriculum. As a hard constraint no clash should occur for the scheduled courses of a student. Similarly, a teacher (or a lecturer) clash must not be allowed. The availability of teachers must be considered. PSTPs form another subset of CTPs that carry many similarities with the high-school course timetabling problems, since the students in a university examination preparation school are in fact high school students. Moreover, the offered courses are arranged according to their curriculums. There are exact and inexact approaches for solving the high-school timetabling problems. The exact methods guarantee to find a solution, such as graph search, while inexact approaches seek for the best solution which is feasible having no hard constraint violations and lowest possible soft constraint violations.

Most of the single point based search metaheuristics applied to high school course timetabling are based on either tabu search or simulated annealing. Abramson (1991) [15] used simulated annealing for course timetabling and proposes a simple parallel algorithm. Herz (1992) [16] utilised tabu search for obtaining course schedules. Schaerf (1996) [17] tested tabu search for high-school course timetabling on different school data using a tool with an interactive interface. Abramson, Dang and Krisnamoorthy (1999) [18] compared different cooling schedules used in simulated annealing for course timetabling. Marte (2004) [19] utilised a constraint programming approach for solving a timetabling problem at German secondary schools of Gymnasium type. Jakobsen, Bortfeld and Gehring (2006) [20] compared their tabu search algorithm to the results obtained during this previous study. Both approaches have a matching performance considering the share of instances solved to feasibility. A hyper-heuristic performs a search based on a set of heuristics. Most of the existing selection hyper-heuristics combine two decision making strategies. A heuristic selection mechanism is used for choosing a heuristic from a lower set of heuristics, while an acceptance mechanism is used for deciding whether to accept or reject a candidate solution [21], [22]. Burke et al. (2007) [23] suggested a tabu-search hyper-heuristic that operated on top of five low level graph heuristics, such as, largest degree first to construct timetables. The representation scheme allowed compaction. Each selected low level heuristic was used to schedule a fixed number of events. The experimental results over a variety of timetabling problems, including course timetabling, showed that the approach had a good average performance.

There are many studies which apply population based metaheuristics, particularly memetic algorithms to high school course timetabling. Ross et al. (1994) [24] and [25] proposed a fast evolutionary algorithm for timetabling. A set of violation directed mutation operators based on selecting a gene to mutate and an allele to mutate to, for genetic algorithms were tested on a set of real and syntactic data for lecture and examination timetabling. Their tests showed that the random selection of a gene and then the selection of an allele by using tournament performed the best. This study extends the idea behind the violation directed mutation to design an effective hill climber using multiple operators given the underlying structure of a problem. Erben and Keppler (1995) [26] generated a weekly timetable for a heavily constraint problem instance using Genetic Algorithms with intelligent operators and binary encoding as a representation. Colorni, et al. (1998) [27] compared various metaheuristics based on GA, simulated annealing and tabu search using an Italian high-school data. Their results indicate that GAs combined with local search is promising. Filho and Lorena (2001) [28] considered a timetabling problem as a clustering problem and applied a modified genetic algorithm, named as constructive genetic algorithm for solving timetabling problems of public schools in Brazil. Wilke et al. (2002) [29] proposed a hybrid genetic algorithm using multiple repair operators and a parameter configuration strategy that operated whenever the search process stagnated. This genetic algorithm included multiple crossover and mutation operators. Any parameter value or operator was chosen randomly whenever needed. The authors reported that this algorithm performed better than the generic genetic algorithm over a large German high school problem instance. Beligiannis et al. (2008) [30] applied a genetic algorithm with multiple mutation operators to a Greek school course timetabling problem. The authors observed that the crossover operator was not as effective as expected, so they discarded it. GA performed better than some previously proposed approaches; column generation and constraint programming. Raghavjee and Pillay (2008) [31] compared the performance of a genetic algorithm, neural network, simulated annealing, tabu search and greedy search on five problem instances. The results showed that GA delivers either a matching performance or it was superior. Raghavjee and Pillay (2010) [32] introduced a South African high school course timetabling problem and used a two-stage genetic algorithm hybridised with a hill climber for solving it. In the first stage, the algorithm aims to satisfy the hard constraints, while in the last stage, the genetic algorithm aims to reduce the soft constraint violations. A different hill climber was used after mutation at each stage.

Most of the genetic algorithms described in this section can be considered as memetic algorithms, since they are all hybridised with either a hill climber or a repair operator that improves a solution, acting effectively as a hill climber. Comparisons to the other approaches indicate that memetic algorithms are successful approaches in high school course timetabling. None of the previous approaches discussed in here attempts to exploit the underlying (hierarchical) structure of the problem. Moreover, the experimental data used in almost all previous studies contained less than hundred events to be scheduled. For example, there were at most 13 classes to be scheduled into 35 teaching hours (time-slots) and 35 teachers in the data set used by Beligiannis et al. (2008) [30]. In this study, we describe a memetic algorithm to solve a course timetabling problem containing more than 1100 events to be scheduled into 96 time-slots and at least 41 teachers (see Section 6.1).

The literature review shows that the researchers working in the area of high school course timetabling do not have an agreement on a set of benchmark instances to compare the performance of approaches. Some of the data sets are publicly available (e.g., [30] and), but we still cannot compare the performance of our approach to the previously proposed ones, even indirectly discarding the differences in the experimental design. This is simply because these data sets contain flat set of events ignoring any (hierarchical) structure for timetabling. Competitions, such as, ITC2007

(http://www.cs.qub.ac.uk/itc2007/) [34], take an active role in setting the state of the art for different problems including course timetabling. Two tracks of this competition covered two different CTPs: post enrollment course timetabling and curriculum based course timetabling and a third track covered the examination timetabling problem. Bonutti et al. [35] describe formulations for the curriculum based course timetabling problem introducing new problem instances additional to the ones provided in the competition. The best performing curriculum based course timetabling and examination timetabling approach turned out to be a multi-stage hybrid algorithm as described in Müller (2009) [36]. First, a feasible solution is constructed, then a hill climbing algorithm is invoked for improving this initial solution. Whenever there is no improvement, great deluge with oscillation and simulated annealing with reheating are invoked successively. Kingston (2010) [37] presents a set of algorithms for room and teacher assignment based on a bipartite matching model in high school course timetabling. More on educational timetabling and recent contributions can be found in the PATAT conference series:
http://www.asap.cs.nott.ac.uk/patat/patat-index.shtml.

ITC2007 did not include high school course timetabling problem. The post enrollment course timetabling and examination timetabling data files are flat, containing no information about the underlying structure of the problem, consequently our algorithm cannot be tested on them. PSTP can be considered as a curriculum based course timetabling problem with different constraints. For example, ITC2007 problem formulation for the curriculum based course timetabling does not consider load fairness (maximum and minimum load per day) for lecturers (teachers), whereas this is a vital soft constraint in PSTP (see Section 3). Also, ITC2007 contains $MinimumWorkingDays$ as a soft constraint to assure that the lectures of each course spreads into a minimum number of days and the closest equivalent constraint is a hard constraint in PSTP. On the other hand, PSTP does not deal with any resource constraints, whereas ITC2007 curriculum based course timetabling problem contains many room constraints, such as $RoomCapacity$ to assure that the number of students taking a lecture in a room does not exceed its capacity. Another observation is that from the point of view of a student, the timetable at a high-school is tightly packed as compared to the one at a university. Because of these distinctive constraints and problem features, our solver cannot be applied to the ITC2007 curriculum based course timetabling problem as it is. This study shows that some algorithms can usefully exploit the underlying structure of a problem. We are hoping that the real world high school course timetabling as well as the examination timetabling benchmarks preserving the hierarchical structure will be included in the next competition.

3. University examination preparation school timetabling problem (PSTP)

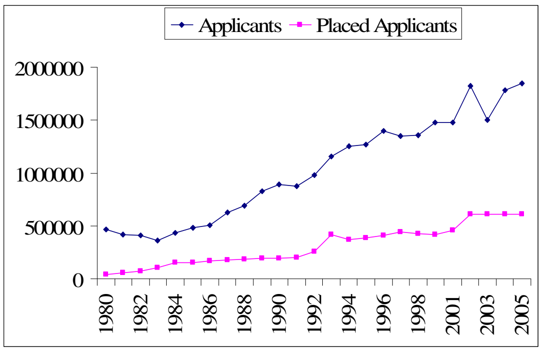


Figure 2: The number of OSS applicants and the placed students versus years

High-school graduates in Turkey enter a country-wide annual Student Selection Examination (OSS), in order to obtain admission to a higher education program at a university. The students are placed into the higher education programs according to their OSS score and set of choices. Less than 40% of the applicants can continue their education in an undergraduate degree program as illustrated in Figure 2. This data is obtained from the website (http://www.osym.gov.tr) of the Student Selection and Placement Center (OSYM) that has been providing a centralized system for fair access and placement of students to higher education programs since 1974, and currently administers the OSS. This situation causes a fierce competition among students, and so there are many private OSS preparation schools in Turkey established as a support mechanism for the candidates. Such schools need to handle the difficult problem of scheduling the course meetings properly subject to a set of constraints.

The organisational and hierarchical structure of the OSS and the school has consequences for the representations and algorithms used to solve the problem. Indeed, in the next section, our ICMA explicitly makes use of this structure. We believe that explicitly using such known structure will be often better than leaving an algorithm to compute it. Hence, here we describe the system and organisation in some detail.

The OSS is a single stage exam having two parts, in which two aptitudes of the entering candidates are assessed: verbal and quantitative. Computing the score of a candidate's exam requires some transformations of the raw result by taking into account the grade-point average of the student at school, the difficulty level of each question that is determined statistically, etc. After the transformations are applied, three different composite *score types* can be computed: *verbal*, *quantitative* and *equally weighted* OSS scores. One of these scores is used in the selection of those candidates who will be considered for the placement to the undergraduate programs. Each department at a Turkish university admits students according to a specific OSS score type. For example, a computer engineering department accepts students based on their quantitative OSS scores. Hence, the students aim to maximize one of those OSS score types to get admitted to a department according to their wish at a university by correctly answering the relevant questions. The private preparation schools (PPSs) act as a support mechanism for the students who will enter the OSS. A student can be admitted to a PPS at any time during his/her high-school education. The high-school education has been extended from 3 to 4 years in Turkey.

In a PPS that has several *branches* at different locations, there are teachers that circulate around these branches, where each registered student attends a set of courses at a specific branch. A student gets prepared to maximize one of the scores, verbal, quantitative or equally-weighted. A *division* indicates the score type that a student aims to collect during OSS. The curriculum of a verbal second-grade (year) high-school student in a PPS differs from that of a verbal or quantitative third-grade high-school student. Hence, the set of courses offered for each student differs according to his/her division and grade. At a branch, depending on the number of registered students, several sections might be arranged to cover them all. Depending on the high-school, the classes might be held before noon, in the afternoon or during the whole day. Moreover, some OSS applicants are high-school graduates. Consequently, PPSs have to arrange course meetings accordingly. For example, eight different sections might be required for all quantitative second-grade high-school students. Some of these sections might require the courses to be scheduled in the morning and some of them in the afternoon during the weekdays. So, all grades are further divided into grade sections. The third-year high-school students that are in the quantitative division must take mathematics, natural sciences and Turkish language courses. All the students that are in the verbal or equally weighted divisions must take some courses in social sciences such as geography and history as well as mathematics and Turkish language courses. But, the number and length of the meetings that must be assigned to the grade sections (of different divisions) can differ. For example, a student in a grade section of an equally weighted division must attend four meetings of the geography course, whereas a student in a grade section of a verbal division must attend six meetings of the geography course. Each grade section groups a set of course sections that a student must attend. A course section denotes a set of meetings for a course. As an example. "BIOL.02[2+3]" might represent the 2nd section of a biology course, and that consists of two meetings of 2 and 3 hours each. The same teacher teaches both meetings. The resulting hierarchical system and logical organisation is illustrated in the top part of Figure 3.

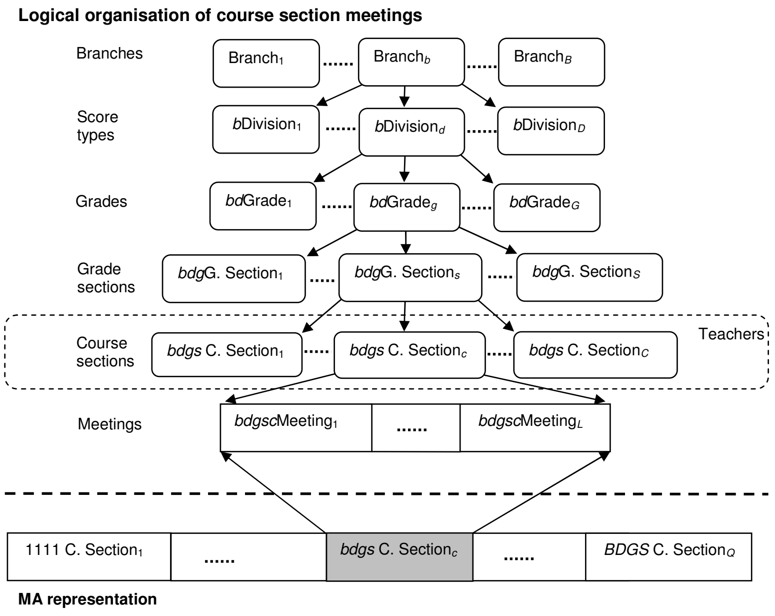


Figure 3: The logical organisation of the course (section) meetings in a PSTP and the representation used in the MAs, where "$bdsg\ C.\ Section_c$" denotes all course meetings of the c^{th} course section of the s^{th} grade section from the g^{th} grade of the d^{th} division in the b^{th} branch of a school.

In a University Exam Preparation School Course Timetabling Problem

(PSTP), the events are the meetings of the course sections. All events must be assigned to some period of p available, within a timetable of d days and t time-slots per day. Note that, the assignment of teachers to events is given by the school, and so is not part of the search problem. Also, the sections are designed by the schools so as to fit within the rooms available, hence there are no capacity constraints: The sizes of events and rooms are not needed. However, there are still strong constraints arising from potential clashes between events, and from the teachers and administration.

Most of the teachers in a PPS are part time having preferences that are treated as hard constraints. There can also be inexperienced teachers who are organized to enter some courses together with the experienced teachers. Each teacher is assigned to a course section beforehand and can take responsibility for more than one course-section. Some of these course sections might be held in different branches. Hence a solution for the PSTP is an assignment for every $x \in E$ (set of events, the course-section meetings) to some $y \in T$ (domain, set of periods), meaning "Course section meeting x starts at time y".

Any acceptable solution must satisfy the following hard constraints:

- **H01.** Meetings from within a grade section cannot overlap. (Teacher and students must attend all the meetings of a grade section).
- **H02.** Meetings of a teacher cannot overlap.
- **H03.** For each course section, the associated meetings must be assigned to different days from each other. That is, within a course section, we cannot have more than one meeting per day.
- **H04.** Teachers can express availability requirements for certain days and/or periods, and these must be respected.
- **H05.** The administration can have requirements for certain days and/or periods for some course meetings, and these must be respected.
- **H06.** Each grade section excludes some periods from before or after noon, and these exclusions must be satisfied.
- **H07.** Some course meetings can be required to be scheduled at the same time as some other event, and these 'sameTimeAs' constraints must be satisfied

- **H08.** Sometimes more than one teacher is given as being assigned to a section of a course, and the resulting non-overlap, and other requirements, must be satisfied.

(In practice, only a small number, less than ten or so of the H05, H07 and H08 constraints as a total appear in a problem instance.)
In addition there are two soft constraints:

- **S01.** The administration specifies a preferred maximum gap in the daily timetables of teachers. A gap for a teacher being an unassigned number of periods between two assigned periods. Gaps over the maximum are penalised, because they are inconvenient and cause the teacher to waste time by having to wait around for the next meeting, and so are not liked.

- **S02.** There are given preferred minimum and a maximum load per day for teachers, and loads outside these bounds are penalised.

Solving the PSTP requires the generation of a assignment to all events, satisfying all the hard constraints and minimimising the total number of soft constraint violations.

4. General ICMA motivations and methods

Genetic and memetic algorithms have successfully used in solving many difficult search and optimization problems, including real-world timetabling problems ([3], [38], [39], [40], [45], [46]). On practical problems, the usual goal is that reasonable amount of time should be spent on obtaining a reasonable schedule for the timetabling of large problem instances, and this motivated Carter (1983) [47] to use divide-and-conquer approach. Traditional approaches such as integer programming are suggested for solving sufficiently small problem instances during the conquering step. Burke and Petrovic (2002) [48] discussed the potential in applying a multistage approach for solving timetabling problems and more.

There are some approaches making use of decomposition in a single stage setting. Qu et al. [41] described an adaptive method which decomposed the events into two subsets as easy and difficult (to schedule) and reordered the examinations in the easy set for constructing high quality timetables. Rahmna et al. [42] investigated the performance of different decomposition and event ordering methods in a similar framework. The decomposition methods in these studies are not based on an underlying static substructure of the problem.

Meisels et al. [50] describe a decomposition approach based on priority functions for assigning teachers to classes subject to mutual exclusion constraints only. A solution was represented in terms of grids which were decomposed into smaller and easier to solve subgraphs (binary Constraint Satisfaction Problem instances). This approach can be considered as a multistage approach in which the stages overlap, since the algorithm backtracks if a consistent solution is not achieved for a given subgraph. The authors, observed that good heuristics existed and consistent solutions were found, in general, without backtracking.

In a recent study, De Causmaecker et al. [49] described a multistage approach in which they used a complete candidate solutions during the search process for solving a university course timetabling problem. A solution is obtained using a subset of constraints at each stage. This solution is fed into the next stage as an initial solution and a new constraint is turned on for the local search algorithm to deal with.

In an other multistage decomposition setting, the data is decomposed into smaller parts and a subset of the data is processed at each stage. McCollum (2006) [51] pointed out that the investigation of approaches involving decomposition is still awaiting attention of the researchers. Qu et al. [43] and Schaerf [44] provides a survey on examination and automated timetabling, respectively. In this section we discuss related multistage approaches and the version that we use. In particular, the hybrid method, ICMA, that adapts this strategy for solving a general timetabling problem is presented.

4.1. Previous multistage strategies

Most of the approaches for timetabling can be converted into a multistage approach based on a strategy that somehow selects and handles a subset of events during each stage. Three different types of such strategies can be identified and are illustrated in Figure 4 and 5. Figure 4 illustrated the first and second strategies that have been used previously.

In the first type, the stages can be arranged such that the approach is applied only to a selected subset of events. Once a satisfactory solution is obtained, based on some criteria only for these events subject to the constraints, the assignment for each event is fixed. Then the next subset is processed as shown in Figure 4(a). Carter (1983) [47] described such a scheme. At each

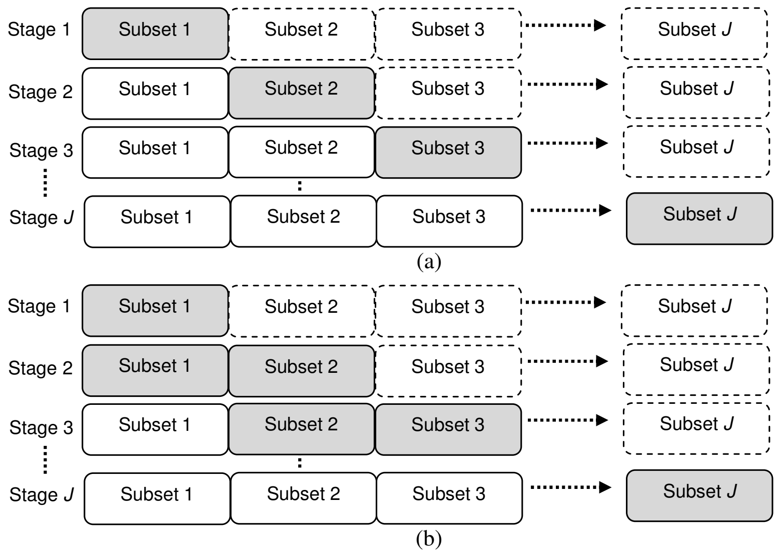


Figure 4: Previously proposed multistage strategies for timetabling. The dashed boxes are the events that are not yet considered. The gray boxes denote the active subset of events on which the algorithm operates and for which it is currently trying to find/improve an assignment. The solid outlined boxes represent the events whose assignment has been found and fixed.

stage, a conventional approach is proposed on a small number of events that will generate a result in a short amount of time. Weare (1995) [33] also investigated this idea for considering different number of events at each stage. This approach has the advantage of only studying a much reduced set of variables at every stage, but also can make some solutions unreachable.

As a second type of strategy, the approach can be applied to the union of an unprocessed subset indicating some unscheduled events and a subset of some previously processed events as illustrated in Figure 4(b). Still, some scheduled events are fixed in this strategy and they are not processed further by the approach. Burke and Newall (1999) [52] tested this strategy by considering pairs of events and then fixing one at each stage using a memetic algorithm. Di Gaspero and Schaerf (2001) [53] compared performances of a constructive heuristic, tabu search approaches and MAs. The multistage MA turned out to generate the best results over four examination timetabling problems.

Neither of these multistage methods have been shown to work for most of the real-world heavily constrained large timetabling problems, such as, nurse rostering, or course timetabling. They both have the disadvantage that whenever a solution to a subset is fixed, some part of the search space is excluded, and so good solutions can be missed.

4.2. Our multistage strategies

Carter (1983) [47] studied a multistage strategy as illustrated in Figure 4(a) within a divide and conquer heuristic. Burke and Newall (1999) [52] investigated another one as illustrated in Figure 4(b) within a memetic algorithm framework. A third strategy within a memetic algorithm framework is used in this work as an extension to the previous studies: No assignment of events is fixed. At each stage, a subset of unscheduled events is incrementally added to be processed by the approach along with the previously processed subset of events as in Figure 5, but the previous assignments remain changeable. The aim of this study is to inquire whether such an incremental multistage approach can provide a better performance as compared to its single stage version.

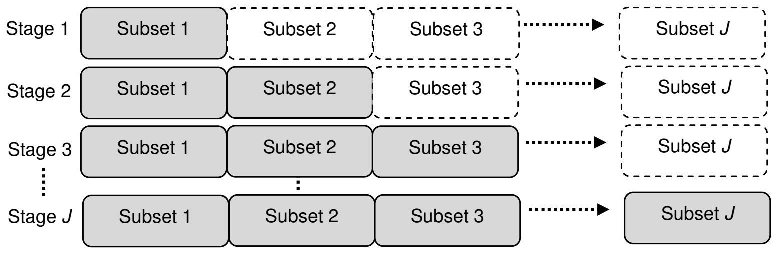


Figure 5: Proposed multistage strategy (see Figure 4 for explanation).

There are two main components of the proposed incremental approach for scheduling events:

- subset selection
- stage termination criteria

4.2.1. Subset selection

Selecting the subset of unscheduled events for addition to the current active set can be done either in a deterministic or randomised method fashion. In a simple case, the size of the added subset can be fixed and the events chosen randomly from the unscheduled events. However, the logical arrangements (groupings) of events in a given problem are very valuable in generating subset selection mechanisms. Consider a term based university course scheduling problem, then there is a subset of courses for each term to be scheduled. As a deterministic mechanism, at each stage, unscheduled courses of a term starting from the first towards the last term can be added for searching a solution. In such a problem, each lecturer is responsible for a subset of courses. As another deterministic strategy, unscheduled courses of a lecturer can be used to increment the number of events to be scheduled based on a specified order. Common events do not cause any problem in any case. For example, there can be two lecturers responsible for a single course. Only the unscheduled events should be added at each stage.

4.2.2. Termination Criteria

After events are added, then the interleaving will run MA until termination criteria cause it to stop. Simple options are a simple limit on the number of iterations at each stage, or achieving an expected solution quality, or a combination. For example, in Burke and Newall (1999) [52], a stage terminates whenever the maximum number of iterations is exceeded.

4.3. General ICMA timetabling

```
Repeat
    Select a group (subset) of new events based on some criteria
    Generate random assignments for the new events in all individuals
    Repeat
        Apply Crossover, Mutation and then Hill Climbing
    Until stage termination criteria are satisfied
Until all events are scheduled and ICMA termination criteria are satisfied
```

Figure 6: Pseudo-code for ICMA for timetabling

Our interleaved multistage method, ICMA, for solving timetabling problems is shown in Figure 6. It might not be possible to timetable all the events, hence the algorithm should still terminate. Also, it might be desirable to run the algorithm even after all events are scheduled for improving the quality of the solutions at hand. ICMA termination criteria control this functionality and for example, a maximum number of iterations could be used as termination criteria. At each stage, a subset of new (unscheduled) events is selected using certain criteria. The assignment for the selected subset of events are randomly generated within a population of candidate solutions. This population is then exposed to the traditional MA operators. Whenever some stage termination criteria is satisfied, then another subset of new events is chosen and again added. This process is repeated until all events are scheduled and some additional termination criteria are satisfied. After the first pass, each individual in the initial population is a partial solution. At each stage, the size of individuals incrementally grows as the new subset of events is added. In this way, no portion of the search landscape is excluded.

A consequence of this scheme is that the genetic operators in the MA must be able to handle the partial solutions. Usually this is straightforward to arrange. At any stage, the genes that are not mapped to any allele do not cause any problem. The fitness computation proceeds as if those events do not exist. Depending on the choice, a gene or a set of genes in a chromosome is allowed to be partially mapped, only if this situation does not disturb the fitness computation. In such cases, genetic operators can ignore the remaining unassigned part. For example, crossover can still exchange the genetic material between mates and mutation can perturb only the mapped portion of the gene.

5. Application of ICMA to the PSTP

In order to show how ICMA is applied to the specific case of PSTP, there are three main components to describe:

A. The MA itself.

B. The construction steps that give the movement to the next stage by the inclusion of more events.

C. The VDHC aspect that controls the application of the constraint-directed hill-climbers within the MA.

and we treat each in turn.

5.1. The MA itself

As an interleaved multistage approach, an MA is used for solving PSTPs as illustrated in Figure 4. An individual in the population contains all (course-section) meetings to be scheduled and its physical implementation reflects the same logical arrangement in Figure 3. A gene corresponds to a course section in the representation used. The initial population is randomly generated in a particular way. The meetings of a given course section are assigned to random timeslots, assuring that each meeting is scheduled on a randomly selected separate day for an individual. Similarly, mutation randomly reschedules the meetings of a course section with no day clash using a probability of 0.25. In this way, the constraint H03 is satisfied at all times. As a mate selection method linear ranking is used, all individuals are sorted with respect to their fitness values. Then a linearly changing rank is assigned based on their position within the population using a selective pressure in the range of [1.0, 5.0]. The probability of choosing the best individual becomes five times larger than choosing the worst individual as a mate from the population. Two crossover operators are implemented forming two new individuals. Modified one-point crossover (m1PTX) swaps parts of selected individuals at a selected point. As a crossover point, one of the start points of the grade sections is randomly chosen. A modified uniform crossover (mUX) exchanges course meetings in each grade section as a whole with a probability of 0.5. Such promising operators derived from the logical arrangement of events are proposed and tested by Alkan and Özcan (2003) [3], previously.

As is common practice, feasibility is relaxed during the search process itself, though not for the final solutions. However the algorithm is such that only H01 and H02 are violated during search, H03-H08 can be preserved during the ICMA process. Hence, the fitness function is a weighted sum of the number of all constraint violations as shown in Equation 1:

$$f(S) = \sum_{i \in \{H01, H02, S01, S02\}} viol(S, i) * w_i \tag{1}$$

where S represents a candidate solution, $viol(S,i)$ counts the violations S due to the constraint i and w_i is the weight for $viol(S,i)$. Each conflict counts as one violation in H01 and H02, while the number of violations is the total deviation from the limits for S01 and S02. For example, if a teacher has a load of 1 in a day and the minimum load is per day is 3; this generates a violation of 2.

5.2. The construction step

In the constructive step, all grade sections receive an additional selected course section meeting simultaneously at a stage. Notice, that this intrinsically exploits the structure of the relationships between meetings. One could have just 'compiled down' the problem so as to only contain constraints between sets of meetings, however, this would have made it difficult to exploit the overall structure during the construction step.

A deterministic mechanism is used as a subset selection method. Largest degree first heuristic [54] is adapted for choosing a course meeting to be added to each grade section. The selection within a grade section is performed using a standard heuristic, but the way it works evenly on all grade sections is important, and novel. The course section with the largest number of meetings and in case of equality with the largest number of duration (largest degree) in each grade section is determined. Then the corresponding course meeting is added to each grade section. For example, assume that in the first grade section, following course section meetings are to be scheduled; BIOL.01[3+2], GEO.01[2+2], MATH.01[4+3+2]. During the stage transition, the first meeting of MATH.01 with duration of 4 is added to the first grade section, since MATH.01 has three meetings to be scheduled, while the rest of them have two meetings. In the next stage, either the second meeting of MATH.01 or the first meeting of BIOL.01 with duration 3 will be added. A random decision is made for such a case. The number of events added at the beginning of a stage is parameterized with respect to the number of grade sections. This parameter is referred to as the *growth factor* (gf). For example, if there are 30 grade sections and gf is 2, then 2 events are added to each grade section yielding an increment of 60 in the overall size of a candidate solution. Due to the choice of subset selection mechanism and the definition of a gene, partially mapped genes occur within the individuals during evolution. Both of the genetic operators handle the partially mapped genes. Moreover, the fitness of an individual can still be computed. The MA at a stage terminates whenever a ratio of all hard constraints (H01-H08) are satisfied by an individual or a maximum number of steps is exceeded. Then the next stage starts. This ratio is referred to as *violation elimination ratio*. As an example, the value of 0.5 indicates that if 50% of all hard constraint violations in an individual are eliminated, then the current stage ends.

5.3. The VDHC aspects

For H07, the course sections that should be scheduled to the same periods have just a single gene in the representation pointed by them. Similarly, each course section that will be taught by more than one teacher has a single gene pointed by the related teachers for H08. The relevant violations are detected by using the list of each teacher and course section assignments. No assignment, outside of the restricted domain of each course meeting forced by H04-H06 is allowed during the evolution.

The MAs use a set of constraint-based hill climbers for H01, H02, S01 and S02. Each hill climber attempts to remove the violations due to a constraint by random rescheduling. A limited number of possibilities are tested and the best one is accepted. The hill climber (repair operator) for H01 is applied after all genetic operators. None of the course meetings in a grade section is allowed to overlap by employing the hill climber for H01.

```
Input: Candidate Solution S
 0. I:= 1
 1. Repeat
 2.   If (no violations of any constraint)
 3.         break
 4.   Select a constraint type i based on the individual violations in S
 5.   Form a list L:= [ course meetings that violates the selected
 6.                     constraint type in S ]
 7.   If (L is not empty)
 8.         Shuffle L randomly
 9.         Repeat
10.               Remove the next course meeting e from L
11.               Apply the constraint oriented hill climber HC_i to e
12.               I:= I + 1
13.               If (no overall improvement in the quality of S)
14.                     Set the assignment of e to its previous value
15.         Until (there is improvement) or (L is empty) or (I>upperBound)
16.   If (L is empty) and (no overall improvement in the quality of S)
17.         Mutate a randomly selected course meeting
18. Until (I>upperBound)
```

Figure 7: Pseudo-code of the violation directed hierarchical hill climbing utilised within the MA

The idea of using a violation directed mutation operator as a genetic algorithm is tested over a set of real and syntactic data for lecture and examination timetabling in [24], [25]. Their results show that the random selection of a gene and then the selection of an allele to be assigned by using tournament perform the best. The tournament strategy favors the assignment that produces the smallest number of violations. [3], [6] and [5] extend this idea for designing different types of operators, including hill climbers to be used in metaheuristics.

In this study, a similar heuristic to the one described in [5] is used to manage the hill climbers for H02, S01 and S02 as a single hill climber (Figure 7). In the first phase, the number of violations due to a constraint type is computed and a hill climber is randomly selected using a ranking strategy (with a virility of 5) based on this information (Line 4). The hill climber selection process is similar to the mate selection process. Instead of the fitness of individuals, the number of violations due to the constraint types is evaluated for selection. It is more likely that the constraint type causing more violations will be selected; hence, the relevant hill climber will be invoked (Line 11). Each hill climber aims to correct the violations of the related constraint type.

After deciding on the constraint type to be repaired, a new phase starts. A meeting should be chosen for removing the selected constraint type violations, hence a list of events is formed that violates the selected constraint type (Lines 5-6). A random permutation of these events is formed (Line 8) and each event is processed consecutively by the hill climber as long as the solution does not improve without exceeding the maximum number of steps (Lines 9-15). If an improving move is detected (Line 15) before reaching the end of the list, then the hill climber goes to the first phase again and the same steps are repeated. If all items in the list are processed and no improvement in the quality of the candidate solution is detected, then mutation is invoked for a randomly selected event from the list (Lines 15-16). The hill climber terminates whenever there is no violations of any constraint left (Lines 2-3) or a maximum number of steps are exceeded (Line 18).

6. Computational Results

Pentium IV 3 GHz. windows machines having 2 Gb of memory are used during the experiments. All runs are repeated fifty times. Initially, some parametric fine tuning experiments are held. Then, ICMA is compared to the conventional MA, which attempts to schedule all course meetings simultaneously. For a fair comparison between the approaches, the experiments are terminated if the expected global optimum is achieved or the execution time exceeds 600 CPU seconds (unless it is mentioned), considering that MA operates on the complete solutions whereas ICMA on the partial solutions. If there are no constraint violations, 0 fitness value is expected. Equal weights are used within the fitness function. As a replacement strategy, the best two individuals in a generation are passed to the next one. The rest of the population is generated using the genetic operators. *Success rate* (s.r.) indicates the ratio of successful runs, achieving the expected fitness to the total number of runs. The success rate, average number of violations and ranking based on these measures are used as performance comparison criteria.

6.1. Experimental data

A real data obtained from Final Dershanesi, a private PPS is used during the experiments. This data referred to as fdm11 is composed of four smaller subsets. Ten more problem instances are formed by concatenating the subsets in different ways as summarized in Table 1. All the problem instances having "fdm" prefix are modified from the raw data, in which each instance is denoted by "fd" prefix. Some events from the raw data are discarded that causes obvious constraint violations. It is more likely that there might be an optimal solution for the modified data as compared to the raw data. Both the raw and modified data sets can be reached from
http://cse.yeditepe.edu.tr/∼eozcan/research/TTML/.

Three sets of preliminary experiments are performed for fine tuning ICMA using the first six problem instances (fdm1-fdm6). Each problem instance requires an optimal schedule to be generated for more than 1100 course meetings. There are 8 days and 12 hours per day in the timetable. For most of the courses, one hour is required for each meeting. The students in a grade section attend 45 to 48 course meetings. Each course section requires 2 to 8 course meetings. The interleave in S01 is fixed as two. The minimum and maximum load of a teacher imposed by S02 is set to 2 and 6 hours per day, respectively. At each stage, the number of events added is the number of grade sections for each problem.

6.2. Preliminary experiments using ICMA

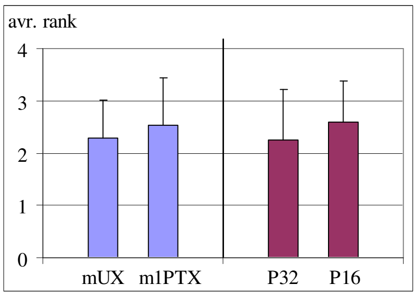
The performances of crossover operator (m1PTX, mUX) and population size (16, 32) combinations within ICMA are tested first (Figure 8). Lower rank denotes a better value for either the best or mean number of violations. Ranks are computed by taking ties into account. The violation elimination ratio is set as 1.0, while the growth factor is set to 1 during the first set of experiments. The maximum number of generations is fixed as 50 for each stage. These choices are arbitrary. It is observed that the mUX crossover is slightly better than m1PTX on average considering the best (offline performance) and average best results (online performance). On the other hand, a population size of 16 is a slightly better choice considering the average best results, but a population size of 32 is a slightly better choice considering the best results. A pair-wise student's t-test comparing the effect of the crossover operator and population size choices on the average performance of MAs for each problem instance shows that there is no statistically significant performance variation between them. For this reason, mUX is chosen as the crossover and a population size of 16 is used during the further experiments. The rest of the settings are kept the same.

A set of values {0, 0.25, 0.5, 0.75, 1.0} for violation elimination ratio are tested, secondly. Performance rank of each ratio based on the best number of violations in 50 runs for each problem instance is provided in Table 2. As the ratio grows, the performance of ICMA gets better in almost all problem instances. The experimental results indicate that aiming to satisfy as many hard constraints as possible is vital as a stage transition criterion. For cases in which this is not possible, having a limit on the maximum number of iterations between stages is sufficient as a termination criterion. The violation elimination ratio is set to 1.0 during the rest of the experiments.

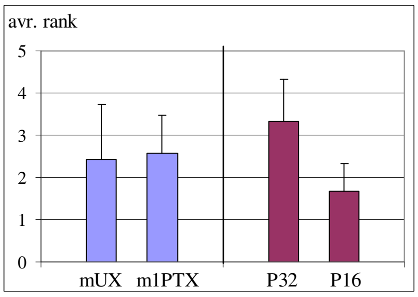
A single course meeting is added to each grade section in the previous experiments. During the last set of experiments, ICMA is tested on fdm1-fdm6 using different growth factors; {1, 2, 3, 4}. Figure 9 summarizes the results. Different growth factors might deliver different performances. For fdm3 and fdm5, a growth factor of 4 and 3 generate the best performance, respectively, while a growth rate of 2 provides either the best performance or a matching performance to the best for the rest of the problem instances. For fdm4, an optimal result cannot be achieved in none of the runs and for any growth factor. Adding two course meetings to a grade section performs the best with an average success rate of 0.31 over the problem instances. Hence, a growth factor of 2 is used in the subsequent experiments.

6.3. Comparison of the conventional MA and ICMA

The duration as a termination criterion is set to 600, 900 and 1200 seconds for the problem instances having a grade section of 30, 45 and 60, respectively. The performance comparison of the conventional MA and ICMA is presented in Table 3. A success rate, in a way, indicates the probability of obtaining a violation free schedule in a single run for the problem. ICMA performs better than the conventional MA in all cases, except for fdm8, considering the success rate and the average number of violations of the best individuals over the runs. A violation free schedule can be obtained for seven out of eleven



(a)



(b)

Figure 8: Performance comparison of crossover operators and population size parameters within ICMA considering (a) the best and (b) mean number of violations over fifty runs. The average values and their standard deviations are obtained by averaging all ranks over all experiments in which two crossover and two population size combinations are used on all problems.

problem instances. ICMA delivers the best average number of violations for the rest of the problems; namely, fdm4, fdm7, fdm10 and fdm11.

Figure 10 shows how the number of violations of the best individual in a population changes during the evolutionary process as an example. Some stage transitions might occur before the maximum number of generations between successive stages is exceeded. That is, the hard constraints might be satisfied less than 50 generations. This indicates the power of the hill climber utilised in the MAs. The mutational component within the proposed hill climber seems to be useful in both MAs. It kicks in whenever the hill climber gets stuck and can not generate an improved solution by considering the selected constraint oriented neighborhood.

The experiments are repeated using the raw data fd1-fd6 and tightening the S01 constraint. The rest of the settings are kept the same. The

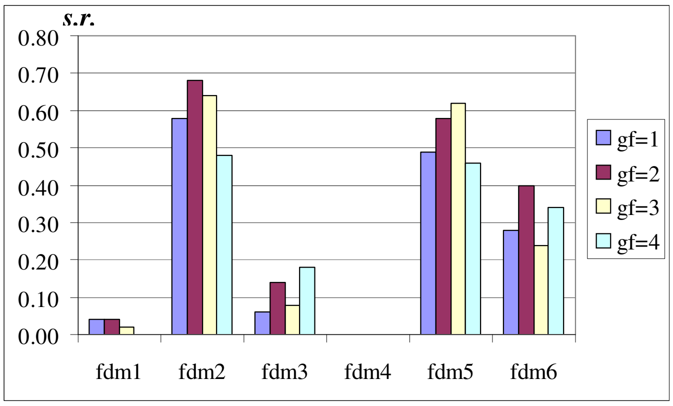


Figure 9: Performance comparison of ICMA based on success rates ($s.r.$) for various growth factors (gf) and for the problem instances fdm1 to fdm6.

results are presented in Table 4. The average performance comparison between the conventional MA and ICMA based on the student's t-test and the %-improvement of the best approach over the other one using the average number of violations for a given problem instance are provided in Table 5. It is observed that ICMA delivers a better average performance when compared to the conventional MA for all problem instances (fd1-6). This performance variation is statistically significant for fd1, fd3, fd4 and fd6. Considering the best performance of MAs in fifty runs, still, ICMA performs better than the conventional MA in all cases, if ties are broken based on the average number of violation and generations. For none of the problem instances all constraints are resolved, except for fd5. For fd2, ICMA achieves the same quality solution as the conventional one by visiting less number of states on average.

7. Conclusion

A new course timetabling problem is presented in this paper: University Exam Preparation School Course Timetabling Problem (PSTP), and a set of such problem instances are made available, each requiring an optimal schedule for more than 1100 course meetings subject to a set of hard and soft constraints.

We have also presented a new hybrid, the "Interleaved Constructive Memetic Algorithm". This works on an active subset of the events, and uses an interleaving of

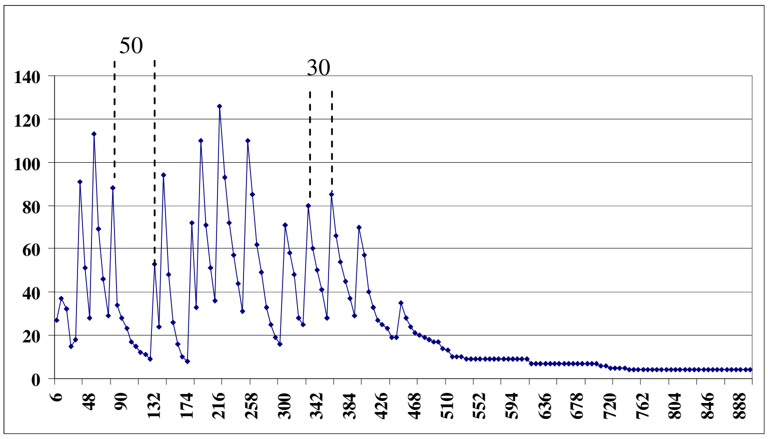


Figure 10: Number of violations versus generation plot: A sample run of ICMA for solving fdm11 showing the number of violations (y-axis) of the best individual at each generation (x-axis).

- A constructive step in which a selection of the currently un-assigned events are to the active subset and are given initial assignments. This selection of the events is done heuristically, and exploits the structure of the problem domain. The novelty of the selection process is in the way it works which enforces fairness of progress across all same sub-structures.

- The memetic algorithm, with its set of constraint directed hill-climbers, is allowed to work on the active set of events so as to improve the assignments they have been given. It is permitted to change all the assignments if it desires as a novel feature which extends the previously proposed strategies in [47] and [52].

On the PSTP, the empirical comparison between the ICMA and its conventional version shows that ICMA achieves better results: Both use the same operators, including the single hill climber; a heuristic that decides the most appropriate hill climber to apply whenever necessary from a set of constraint based hill climbers. We also note that converting from MA to ICMA is relatively easy. Although applied here to a particular course timetabling problem, the proposed incremental strategy is a general strategy that can be adapted easily by the existing approaches for timetabling and scheduling. There is already strong evidence that some examination timetabling [7], workforce scheduling, particularly nurse rostering [5], [6] problems are highly hierarchically structured which can be exploited by our method.

Comparing ICMA with other meta-heuristics the primary relevant difference is that ICMA exploits the explicitly-given hierarchical structure. Quite possibly other metaheuristics can also exploit that structure and improve their performance, however, direct empirical comparison is often stymied by the general propensity for benchmark problems to discard such hierarchical structure. Note that it is possible that other standard meta-heuristics would outperform ICMA without using the structure, but even in that case the point made by this study is that extending such meta-heuristics so as to exploit the structure might also improve their performance. This also emphasises that it would be good practice for those converting real instances to benchmark ones to ensure that the structure is preserved.

Naturally, we also intend to explore the performance of ICMA on domains for which structure preserving benchmarks are provided. As a potential example, the structure of the PSTP is similar to those used in the ITC2007 course timetabling track, [35], and so this suggests that ICMA be also tested on this domain. We note that [55] uses an integer programming method (as opposed to a metaheuristic) but that it does exploit the given grouping structure of the events in the problem; thus re-iterating the general potential utility of structure exploitation.

Implementation specific mechanisms for the subset selection and the termination criteria at each stage for the proposed strategy can also be investigated further. Future work will also look into the mechanisms that underlie the ICMA, and the questions raised by this work. For example, it seems reasonable that the interleaving acts as an additional diversification mechanism within the MA due to the newly introduced events and their random scheduling at each stage. Another hypothesis we want to explore arises from the observation that the ICMA gradually increases the size of the active set of events. When the active set is small, then search effort is probably not useful as the problem is too easy. However, if the active set is too large (or includes all events as in the usual MA) then it is possible that search effort is not efficiently used as the problem becomes too constrained to allow easy movement within the search space. Possibly the main effectiveness of the ICMA method arises at intermediate sizes of the active sets; sets that are large enough to be informative but not so large as to cause the search to stagnate. Also possibly related, is the commonly held view that many problems contain a core subproblem that is hard to solve, but once solved then the other variables depend on the core in a fashion that is relatively easy to handle. A potential problem with methods that work on assignments to all variables (such as GAs and MAs) is that they can get confused by the dependent variables. Yet, fixing this by directly identifying the care subproblem is also hard. We hope that ICMA will lead to methods that have a higher chance of focussing on the hard core at some stage during the multi-stage construction, and so can reduce the distracting effects of the easier components of the problem.

Acknowledgment

This research is funded by TUBITAK (The Scientific and Technological Research Council of Turkey) under the grant number 107E027.

References

[1] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," SIAM J. Computing, 5(4), pp. 691-703, 1976.

[2] E. K. Burke, and J. D. Landa Silva, "The design of memetic algorithms for scheduling and timetabling problems," in Studies in Fuzziness and Soft Computing. Recent Advances in Memetic Algorithms and Related Search Technologies, W. Hart, J. Smith and N. Krasnogor (ed.), Springer, vol. 166, pp. 289-311, 2005.

[3] A. Alkan, and E. Özcan, "Memetic algorithms for timetabling," in Proc. of IEEE Congress on Evolutionary Computation, pp. 1796-1802, 2003.

[4] A. Viana J. Pinho de Sousa, M.A. Matos, "GRASP with constraint neighbourhoods - an application to the unit commitment problem," in Proc. of the 5th MIC, 2003.

[5] E. Özcan, "Memes, self-generation and nurse rostering," Lecture Notes in Computer Science 3867, Springer-Verlag, selected papers from the 6th Int. Conf. on the PATAT, pp. 85-104, 2007.

[6] E. Özcan, "Memetic algorithms for nurse rostering," P. Yolum (Eds.): Lecture Notes in Computer Science 3733, Springer-Verlag, The 20th ISCIS, pp. 482-492, 2005.

[7] E. Özcan, E. Ersoy, "Final exam scheduler-FES," in Proc. of 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1356-1363, 2005.

[8] E. Ersoy, E. Özcan, Ş. Uyar, "Memetic algorithms and hyperhill-climbers," in Proc. of the 3rd Multidisciplinary Int. Conf. On Scheduling: Theory and Applications, P. Baptiste, G. Kendall, A. M. Kordon, F. Sourd (ed.), pp. 159-166, 2007.

[9] Ender Özcan, Alpay Alkan, "A memetic algorithm for solving a timetabling problem: an incremental strategy," in Proc. of the 3rd Multidisciplinary Int. Conf. On Scheduling: Theory and Applications, P. Baptiste, G. Kendall, A. M. Kordon, F. Sourd (ed.), pp. 394-401, 2007.

[10] E. Özcan, "Towards an XML based standard for timetabling problems: TTML," Multidisciplinary Scheduling: Theory and Applications, Springer Verlag, 163 (24), 2005.

[11] J. H. Holland, "Adaptation in natural and artificial systems," Univ. Mich. Press, 1975.

[12] P. Moscato and M. G. Norman, "A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems," Parallel Computing and Transputer Applications, pp. 177-186, 1992.

[13] Y.S. Ong, A.J. Keane, "Meta-Lamarckian learning in memetic algorithms," IEEE Trans Evol Comp 8(2):99-110, 2004.

[14] Y.S. Ong, M.H. Lim, Z. Ning, K.W. Wong, "Classification of adaptive memetic algorithms: a comparative study," IEEE Trans SMC BC 36(1):141-152, 2006.

[15] D. Abramson, "Constructing school timetables using simulated annealing, sequential and parallel algorithms," Management Science, 37(1), pp. 98-113, 1991.

[16] A. Hertz, "Finding a feasible course schedule using a tabu search," Discrete Applied Mathematics, 35, pp. 255-270, 1992.

[17] A. Schaerf, "Tabu search techniques for large high- school timetabling problems," in Proc. of the Fourteenth National Conference on AI, pp. 363-368, 1996.

[18] D. Abramson, H. Dang, and M. Krisnamoorthy, “Simulated annealing cooling schedules for the school timetabling problem,” Asia- Pacific J. of Op. Res., 16, pp. 1-22, 1999.

[19] M. Marte, “Towards constraint-based school timetabling,” in Proc. of the Workshop on Modelling and Solving Problems with Constraints, ECAI 2004, pp. 140-154, 2004.

[20] F. Jakobsen, A. Bortfeld, H. Gehring, “Timetabling at German secondary schools: Tabu search versus constraint programming,” in Proc. of the 6th Int. Conf. on the PATAT, pp. 439-442, 2006.

[21] B. Bilgin, E. Özcan, E. E. Korkmaz, “An experimental study on hyper-heuristics and exam scheduling,” Selected papers from the International Conference on Practice and Theory of Automated Timetabling 2006, Lecture Notes in Computer Science, vol. 3867, pp. 85-104, 2007.

[22] E. Özcan, B. Bilgin, E.E. Korkmaz, “A comprehensive survey of hyper-heuristics,” Intelligent Data Analysis 12(1):1-21, 2008.

[23] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic and R. Qu, “A graph-based hyper-heuristic for educational timetabling problems,” EJOR, 176(1), pp. 177-192, 2007.

[24] P. Ross, D. Corne, and H-L. Fang, “Improving evolutionary timetabling with delta evaluation and directed mutation,” in Proc. of PPSN III, pp. 556-565, 1994.

[25] P. Ross, D. Corne, and H-L. Fang, “Fast practical evolutionary timetabling,” in Proc. of AISB Workshop on Evolutionary Computation, pp. 250-263, 1994.

[26] W. Erben and J. Keppler, “A genetic algorithm solving a weekly course-timetabling problem,” in Proc. of the First Int. Conf. on the Practice and Theory of Automated Timetabling (ICPTAT), Napier University, Edinburgh, pp. 21-32, 1995.

[27] A. Colorni, M. Dorigo, and V. Maniezzo, “Metaheuristics for high-school timetabling,” Computational Optimisation and Applications, vol. 9, pp. 275-298, 1998.

[28] G. R. Filho, L. A. N. Lorena, "Constructive evolutionary approach to school timetabling," Lecture Notes in Computer Science, Springer, vol. 2037, pp. 130-139, 2001.

[29] P. Wilke, M. Grobner,and N. Oster, "A Hybrid Genetic Algorithm for School Timetabling," Lecture Notes in Computer Science 2557, Springer-Verlag, Advances in Artificial Intelligence, pp. 455-464, 2002.

[30] G. N. Beligiannis, C. N. Moschopoulos, G. P. Kaperonis, and S. D. Likothanassis. 2008. "Applying evolutionary computation to the school timetabling problem: The Greek case," Comput. Oper. Res., 35(4), 1265-1280, 2008.

[31] R. Raghavjee, and N. Pillay, "An application of genetic algorithms to the school timetabling problem," Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: Riding the wave of technology, pp. 193-199, 2008.

[32] R. Raghavjee, and N. Pillay, "An informed genetic algorithm for the high school timetabling problem." in Proc. of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT '10). ACM, New York, NY, USA, pp. 408-412, 2010.

[33] R. F. Weare, "Automated examination timetabling," Ph.D. dissertation, University of Nottingham, Department of Computer Science, 1995.

[34] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A.J. Parkes, L. Gaspero, R. Qu, and E.K. Burke, "Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition," INFORMS Journal on Computing Vol. 22, pp. 120-130, 2010.

[35] A. Bonutti, F. De Cesco, L. Di Gaspero, A. Schaerf, "Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, visualization, and results." Annals of Operations Research, to appear. DOI: 10.1007/s10479-010-0707-0

[36] T. Müller, "ITC2007 solver description: A hybrid approach," Annals of Operations Research 172:429446, 2009.

[37] J.H. Kingston , “Resource assignment in high school timetabling,” Annals of OR, 2010, DOI:10.1007/s10479-010-0695-0.

[38] D. E. Goldberg, “Genetic algorithms in search, optimization, and machine learning,” Addison- Wesley, Reading (MA), 1989.

[39] N. Krasnogor, “Studies on the theory and design space of memetic algorithms,” Ph.D. Thesis, University of the West of England, Bristol, United Kingdom, 2002.

[40] B. Paechter, R. C. Rankin, A. Cumming, and T. C. Fogarty, “Timetabling the classes of an entire university with an evolutionary algorithm,” in Proc. of Parallel Problem Solving from Nature (PPSN V), pp. 865-874, 1998.

[41] R. Qu and E.K. Burke, “Adaptive Decomposition and Construction for Examination Timetabling Problems,” in Proc. of the 3rd Multidisciplinary International Scheduling: Theory and Applications 2007 (MISTA 2007), pp. 418-425, 2007.

[42] S.A. Rahman, A. Bargiela, E. K. Burke, B. McCollum and E. Özcan, “A Construction Approach for Examination Timetabling based on Adaptive Decomposition and Ordering,” in Proc. of the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2010), pp. 353-372, 2010.

[43] R. Qu, E.K. Burke, B. McCollum, L.T.G. Merlot, and S.Y. Lee, “A survey of search methodologies and automated system development for examination timetabling,” Journal of Scheduling, 12(1): 55-89, 2009.

[44] A. Schaerf, “A survey of automated timetabling,” Artif. Intell. Rev., 13(2): 87-127, 1999.

[45] E. Özcan, C. Başaran, “A case study of memetic algorithms for constraint optimization,” Soft Computing: A Fusion of Foundations, Methodologies and Applications, 13(8-9), pp. 871-882, 2009.

[46] E. Özcan, and E. Onbaşıoğlu, “Memetic algorithms for parallel code optimization,” Int. J. on Parallel Processing, vol. 35, no. 1, pp. 33-61, 2007.

[47] M. W. Carter, “A decomposition algorithm for practical timetabling problems,” Dept. of Industrial Engineering, University of Toronto, Working Paper 83-06, April, 1983.

[48] E. K. Burke, S. Petrovic, “Recent Research Directions in Automated Timetabling”, EJOR, 140, pp. 266-280, 2002.

[49] P. De Causmaecker, P. Demeester, G. Vanden Berghe, “A decomposed metaheuristic approach for a real-world university timetabling problem,” European Journal of Operational Research, vol. 195, no. 1, pp. 307-318, 2009.

[50] A. Meisels, J. Ell-Sana, E. Gudes, “Decomposing and solving timetabling constraint Networks,” Computational Intelligence, vol. 13, no. 4, pp. 486-505, 1997.

[51] B. McCollum, “University Timetabling: Bridging the Gap between Research and Practice,” in Proc. of the 6th Int. Conf. on the PATAT, pp. 15-31, 2006.

[52] E. K. Burke, J. P. Newall, “A multistage evolutionary algorithm for the timetable problem,” IEEE Trans. on Evolutionary Computation, vol. 3, no. 1, pp. 63-74, 1999.

[53] L. Di Gaspero, A. Schaerf, “Tabu search techniques for examination timetabling” In E. Burke and W. Erben, editors, Practice and Theory of Automated Timetabling III, no. 2079 in Lecture Notes in Computer Science, pp. 104-117, 2001.

[54] M. W. Carter, G. Laporte, “Recent developments in practical timetabling,” Selected papers from the Second International Conference on Practice and Theory of Automated Timetabling II, Lecture Notes In Computer Science; vol. 1408 , pp. 3-19, 1997.

[55] E.K. Burke, J. Marecek, A.J. Parkes, H. Rudova, “A supernodal formulation of vertex colouring with applications in course timetabling”. Annals of Operational Research, to appear.

Table 1: Characteristics of the experimental data set, where minl and maxl denote the minimum and maximum total load of the teachers for a given problem, respectively.

	No. of	Course	No. of	No. of	No. of	Grade	No. of		
Label	Meetings	Sections	Branches	Divisions	Grades	Sections	Teachers	minl	maxl
fdm1	1183	253	1	3	3	30	48	3	46
fdm2	1125	291	2	3	3	30	49	6	40
fdm3	1123	292	2	3	3	30	50	8	42
fdm4	1156	304	2	2	2	30	42	6	46
fdm5	1154	305	2	2	2	30	42	9	41
fdm6	1096	343	1	2	1	30	41	5	46
fdm7	1732	424	2	4	4	45	50	8	58
fdm8	1730	425	2	4	4	45	50	8	63
fdm9	1672	463	2	4	3	45	50	8	55
fdm10	1703	476	2	3	2	45	42	9	64
fdm11	2279	596	2	4	5	60	50	8	76

Table 2: Performance comparison of ICMA for various violation elimination ratios (v.e.r) based on ranking for each problem instance.

Label \ v.e.r	1	0.75	0.5	0.25	0
fdm1	1	2	3	4	5
fdm2	3	3	3	3	3
fdm3	1	3	3	3	5
fdm4	1	2	3	4	5
fdm5	3	3	3	3	3
fdm6	2	1	5	3.5	3.5
avr.	1.8	2.3	3.3	3.4	4.1

Table 3: Comparison of MAs, where *s.r.*, *avr.* (and *st.d.*), *viol.* and *gen.* denote success rate, average fitness, standard deviation, violations and generations, respectively. For each problem, the bold entry marks the best performing approach; the comparison criterion is the success rate and the average number of violations as tie breaker.

	Conventional MA					*ICMA*				
label	*s.r.*	*avr. viol.*	*st.d.*	*avr. gen.*	*st.d.*	*s.r.*	*avr. viol.*	*st.d.*	*avr. gen.*	*st.d.*
fdm1	0.28	2	2.59	1067	315.94	**0.40**	1	1.89	886	336.19
fdm2	0.62	0	1.38	735	417.32	**0.68**	0	1.08	660	350.85
fdm3	0.10	3	2.59	1158	207.8	**0.18**	2	1.82	1053	126
fdm4	0	15	4.77	1312	45.72	0	**10**	4.19	1230	37.71
fdm5	0.56	0	0.65	828	486.41	**0.58**	0	0.65	798	413.29
fdm6	0.02	5	3.11	1339	73.65	**0.04**	5	2.75	1263	60.19
fdm7	0	12	4.64	1531	52.26	0	**7**	3.51	1404	40.5
fdm8	**0.44**	1	2.02	1090	511.44	0.36	1	1.51	1069	352.3
fdm9	0.14	5	4.64	1486	158.96	**0.18**	3	3.43	1370	190.75
fdm10	0	13	3.89	1675	88.91	0	**12**	3.07	1572	60.03
fdm11	0	12	4.85	1802	85.1	0	**8**	3.54	1625	68.23

Table 4: Comparison of MAs, where *s.r.*, *avr.* (and *st.d.*), *viol.* and *gen.* denote success rate, average fitness, standard deviation, violations and generations, respectively. The label of the modified version of each corresponding raw problem instance is provided in parenthesis. For each problem, the bold entry marks the best performing approach; the comparison criterion is the success rate and the average number of violations as tie breaker.

	Conventional MA					*ICMA*				
label	*best*	*avr. viol.*	*st.d.*	*avr. gen.*	*st.d.*	*best*	*avr. viol.*	*st.d.*	*avr. gen.*	*st.d.*
fd1 (fdm6)	39	52	8.4	1264	34.1	**19**	29	5.4	1323	5.4
fd2 (fdm2)	2	5	3.1	1301	36.7	2	**4**	1.9	1152	20.7
fd3 (fdm3)	12	19	4.2	1293	17.8	**7**	13	3	1165	28.6
fd4 (fdm4)	49	69	9.3	1213	34.9	**27**	48	8	1204	25.6
fd5 (fdm5)	0	1	1.1	1031	472.8	0	1	1	**888**	339.9
fd6 (fdm1)	11	20	5	1286	21.5	**2**	10	3.9	1128	34.1

Table 5: Average performance comparison of MAs (see Table 4). X>Y indicates that the approach X performs significantly better than Y within a confidence interval of 95% based on the student's t-test over fifty runs for a given problem, while X≈Y indicates that X performs slightly better than Y and this performance variation is not statistically significant. %-impr. denotes the percentage improvement that X provides over Y based on the average number of violations.

label	*performance*	%-impr.
fd1 (fdm6)	ICMA $>$ *Conventional* MA	79.3
fd2 (fdm2)	ICMA $\approx$ *Conventional* MA	3.4
fd3 (fdm3)	ICMA $>$ *Conventional* MA	20.7
fd4 (fdm4)	ICMA $>$ *Conventional* MA	73.4
fd5 (fdm5)	ICMA $\approx$ *Conventional* MA	0
fd6 (fdm1)	ICMA $>$ *Conventional* MA	34.5