Memetic Algorithms to the High School Timetabling Problem

George H. G. Fonseca
Exact and Applied Sciences Department
Federal University of Ouro Preto
Ouro Preto, Brazil 35400-000
Email: george@decea.ufop.br

Haroldo G. Santos
Computing Department
Federal University of Ouro Preto
Ouro Preto, Brazil 35400-000
Email: haroldo@iceb.ufop.br

***Abstract*—This work presents the application of Memetic Algorithms to the High School Timetabling Problem. The addressed model of the problem was proposed by Third International Timetabling Competition (ITC), which released many instances from educational institutions around the world and attracted seventeen competitors. The Memetic Algorithm uses as subroutine the winner algorithm of the Third ITC in the refinement phase. It consists in apply a mixed Simulated Annealing - Iterated Local Search approach to all the individuals in the population at each iteration. The Memetic Algorithms were able to overcome the winner of Third ITC solver, beating it in 10 out of 16 instances. Most of them were instances with less than 1.000 lessons to schedule, thus, we conclude that the Memetic Algorithm approach is suitable, especially to small instances of the problem.**

I. Introduction

The High School Timetabling Problem is faced by many educational institutions around the world. It basically consists in to assign timeslots for teachers $\times$ class meetings in such way that no teacher or class attends more than one lesson at same time. Generally, this assignment is repeated weekly until the end of semester. Many other constraints can be considered as well, like availability of teachers, to avoid idle times and to limit the number of lessons of the same subject in a day. Beyond its practical importance, this problem was proven to be $\mathcal{NP}$-Hard [1], [2] and progress in solving such problems is a major goal of current research in Operations Research and Artificial Intelligence.

Due to the problem's importance, three international competitions were made to bring scientists with different optimization techniques to compare their solvers in the same environment. The first one happened in 2003 [3] and was won by Kostuch [4] with a 3-phase Simulated Annealing-based [5] approach. The second one [6], occurred in 2007 and, composed by three separated tracks, was won by Muller [7] also with an Simulated Annealing based approach. The last one [8] happened in 2012 and was won by a mixed approach of Simulated Annealing (SA) and Iterated Local Search (ILS) algorithms approach [9].

Results of all ITC competitions show that local search is a key ingredient for producing good solvers to the High School Timetabling Problem. Some evolutionary methods were also proposed in the competitions, but they did not were as successful as the local search methods. Methods based in Integer Programming were also proposed to the problem [10], but they can only solve exactly a subset of instances of the problem in a viable processing time.

In this sense, the main goal of this work is present an evolutionary approach to the ITC2012 model of the High School Timetabling Problem. In our approach, we proposed a memetic algorithm, which can take advantage of the evolutionary approach keeping a local search phase. Our approach showed good results, beating the Third ITC's winner method in many instances.

The remaining of this work is organized as follows: in Section II it will be presented the model of High School Timetabling Problem proposed by the Third ITC. In Section III it will be presented the solution approach. Section IV presents computational experiments made. Finally, in Section V, concluding remarks are presented.

II. High School Timetabling Problem Model

The Third ITC aimed to stimulate the timetabling research generally, and especially to encourage the alignment of research with practice by offering real-world instances of timetabling problems for solution. The organizers also provided a benchmark to adjust processing times and a validator.

Instances were specified in the XHSTT format, which is an XML based format, strictly made to specify timetabling problems. Post [11] also highlight that this format can specify instances of another timetabling problems besides the scholar context. Thus, our solver is able to solve any instance of timetabling problem specified in this format.

The addressed model of High School Timetabling (HST) Problem came up with the goal of providing a generic model capable to address the various features of the High School Timetabling Problem around the world [12] [13] [14] [15] [16] [10] [17]. The model is split in four main entities.

A. Times

The time entity consists of a single timeslot or a set of timeslots, called time group. The timeslots are commonly also grouped by day (e.g. timeslots of Monday).

B. Resources

The resources are divided in three main categories: classes, teachers and rooms [17]:

classes a group of students attends to an event (lesson). Important constraints to the students are control their idle times and the number of lesson by day;

teachers a teacher can be pre-assigned to attend an event. In some cases it is not pre-assigned and should be assigned according to their qualifications and workload limits;

rooms most of events take place on a room. One room has a certain capacity and a set of features.

C. Events

An event is a set of meets (lessons) about a subject. Events may be also grouped into an event group. A timeslot assignment to an event is called meet and a resource assignment to an event is called task. The term class is used to designate a group of students who attend the same set of events. Other kinds of events, like meetings are allowed by the model [17]. An event has the following attributes:

- a *duration* that represents the number of lesson which an event is composed;
- a *course* related to the event;
- the *workload* that will be added to the total workload of resources assigned to the event;
- the *pre-assigned resources* to attend the event (optional);
- the *pre-assigned timeslots* to attend the event (optional).

D. Constraints

Post [17] groups the constraints in three categories: basic constraints of scheduling, constraints to the events and constraints to the resources. The objective function $f(.)$ is calculated in terms of violations to each constraint penalized according to their weight (so this is a minimization problem). They were also divided in hard constraints, whose attendance is mandatory; and soft, whose attendance is desirable but not mandatory. Each instance can define whether a constraint is hard or soft and its weight. For more details, see [17].

1) Basic Constraints of Scheduling:

- ASSIGN TIME. Assign the required number of timeslots to each lesson;
- ASSIGN RESOURCE. Assign the required resources to each lesson;
- PREFER TIMES. Indicates that some event have preference for a particular timeslot;
- PREFER RESOURCES. Indicates that some event have preference for a particular resource.

2) Constraints to Events:

- LINK EVENTS. Schedule a set of events to the same timeslot;
- SPREAD EVENTS. Specify the allowed number of occurrences of an event in time groups between a minimum a maximum number of times. This constraint can be used, for example, to define a daily limit of lessons;
- AVOID SPLIT ASSIGNMENTS. For each event, assign a particular resource to all of his meets. With this constraint, we can enforce all the allocations of an event to lie in the same room;
- DISTRIBUTE SPLIT EVENTS. For each event, assign between a minimum and a maximum meets of a given duration. This constraint is needed since a great number of consecutive lessons of the same subject can prejudice the students' performance;
- SPLIT EVENTS. Limits the number of non-consecutive meets in which an event should be scheduled and his duration. One example of application of this constraint is to enforce an event composed of four lessons to be assigned in two blocks of contiguous lessons. This may be useful when a single lesson is not enough to discuss about a subject.

3) Constraints to Resources:

- AVOID CLASHES. Assign the resources without clashes (i.e. without assign the same resource to more than one lesson at any timeslot);
- AVOID UNAVAILABLE TIMES. Avoid assigning resources on the times that they are not available. With this constraint, it is possible, by example, to avoid that a room be used in a time where it is reserved to cleaning or avoid that a teacher be allocated in a timeslot in which he is student of a capacitation course;
- LIMIT WORKLOAD CONSTRAINT. To each event there is an associated workload. This workload is added to the resources which attend the event. The goal of this constraint is to keep the workload of the resources between a minimum and a maximum bound;
- LIMIT IDLE TIMES. The number of idle times in each time group should lie between a minimum and a maximum bound for each resource; typically, a time group consists of all timeslots of a given day of the week. this constraint is used to avoid times inactive timeslots between active timeslots in the schedule of each resource;
- LIMIT BUSY TIMES. The number of busy times in each day should lie between a minimum and a maximum bound for each resource. A great number of number of allocations in the same day can prejudice the students and teachers performance;
- CLUSTER BUSY TIMES. The number of time groups with a timeslot assigned to a resource should lie between a minimum and a maximum limit; this can be used, for example, to concentrate teacher's activities in as few days as possible.

III. SOLUTION APPROACH

Our approach uses the Kingston High School Timetabling Engine (KHE) [18] to generate an initial population. Afterwards, we have applied a memetic algorithm to improve the

population. The local search phase of our memetic algorithm is quite similar to the local search presented by the ITC2012 winner [9], being composed by a Simulated Annealing - Iterated Local Search method. These elements will be explained in the following subsections.

A. Build Method

The KHE is a platform for handling instances of the addressed problem. It also provides a solver, used to build initial solutions in the presented approach. The KHE's solver was chosen to generate the initial solutions since it is able to find reasonably good solutions very quickly.

The incorporated solver is based on the concept of Hierarchical Timetabling [19], where smaller allocations are joint to generate bigger blocks of allocation until a full representation of the solution is developed. Hierarchical Timetabling is supported by the Layer Tree data structure [19], consisting of nodes that represent the required meet and task allocation. An allocation may appear in at most one node. A Layer is a subset of nodes having the propriety that none of them can be overlapped in time. Commonly, nodes are grouped in a Layer when share resources.

The hard constraints of the problem are modeled to this data structure and then a Matching problem is solved to find the times/resources allocation. The Matching is done by connecting each node to a timeslot or resource respecting the property of Layer. For full details, see [19] [18].

B. Neighborhood Structure

The neighborhood structure $N(s)$ is composed by seven movements. The neighborhood about only one movement k is denoted by $N_k(s)$. The movements are:

1) Lesson Swap (LS). Two lessons l_1 and l_2 are selected and have their timeslots t_1 and t_2 swapped;
2) Lesson Move (LM). A lesson l_1 is moved from its original timeslot t_1 to a new timeslot t_2;
3) Lesson Block Move (LBM). As LS movement, swap the timeslot of two lessons l_1 and l_2, but, when the lessons have different duration, l_1 is moved to the last timeslot occupied by l_2. This movement allow timeslot swaps without lose the contiguity of allocations;
4) Resource Swap (RS). Two lessons e_1 and e_2 have their assigned resources r_1 and r_2 swapped. Resources r_1 and r_2 should play the same role to allow the swap (e.g. both have to be teachers);
5) Resource Move (RM). A lesson l_1 has his assigned resource r_1 replaced by a new resource r_2 between the available resources to attend it;
6) Kempe Move (KM). Two timeslots t_1 and t_2 are selected. Every lesson of t_1 and t_2 are listed and represent a node n in a graph. If two nodes (lessons) n_1 and n_2 in this graph share resources, they are connected with an edge. Edges are created only between nodes assigned in distinct timeslots, thus, the generated graph is a bipartite one, known as conflict graph. Every edge in the conflict graph also has a weight, formed by the cost difference in the objective function assuming the exchange of timeslots between the events depicted in the pair (n_1, n_2). Afterwards, we look up for the path with the lowest cost in the conflict graph and makes up the exchange of timeslots in chain;
7) Permute Resources (PR). A resource r_1 is selected and the permutation of the lessons allocated to it is made [1]. Each permutation generates a new solution. Later, the best solution (with the lower cost) is selected among the obtained permutations.

The movement k in $N(s)$ is randomly selected in order to generate a neighbor. If the instance require the assignment of resources (i.e. there exists at least one ASSIGN RESOURCE constraint), the movements are chosen based on the following probabilities: ES $= 0.20$, EM $= 0.38$, EBM $= 0.10$, RS $= 0.20$, RM $= 0.10$ and KM $= 0.02$. Otherwise, the movements RS and RM are not used and the odds become: ES $= 0.40$, EM $= 0.38$, EBS $= 0.20$ and KM $= 0.02$. Due to its high computational cost, the movement Permute Resources is applied only in the perturbation phase of ILS algorithm. These values were empirically adjusted.

C. Memetic Algoritms

Many metaheuristic techniques have emerged in the last thirty years to solve combinatorial optimization problems [20]. Among these techniques, Genetic Algorithms (GA) and its variation Memetic Algorithms (MA) have been successfully employed in many important and practical timetabling problems [21], [22], [23].

The Genetic Algorithms are a metaheuristic based on an analogy with natural process of evolution. Given a population p, the individuals with the best genetic features have better chances to survive and to produce children increasingly fit, while the less fit individuals tend to disappear [24].

The GA starts with an initial population of n solutions $P = \{s_1, s_2, \ldots, s_n\}$. In each iteration, the individuals of P generate a set of new individuals $R = \{s_{n+1}, s_{n+2}, \ldots, s_{n+m}\}$ by the *Crossover* process; some of them are randomly modified by the *Mutation* process; afterwards, n individuals are *selected* to compose the new generation $P^{'}$ of the population. This process is repeated until a stop condition is reached.

Proposed by [25], Memetic Algorithms are a variation of GAs, in which we submit some individuals to a *Refinement* phase at each iteration before the *Crossover* and *Mutation* phases. The Algorithm 1 presents the developed implementation of Memetic Algorithms. The *Crossover*, *Mutation*, *Selection* and *Refinement* phases will be detailed in the following sections.

1) Crossover: In the Crossover phase of MA, we equally split the population P in two populations $P_1 = \{s_1, s_2, ..., s_{\frac{n}{2}}\}$ and $P_2 = \{s_{\frac{n}{2}+1}, s_{\frac{n}{2}+2}, ..., s_n\}$. Afterwards, we match an individual $s \in P_1$ with the individual $s^{'} \in P_2$ corresponding to the position which s fill in P_1. Thus s and $s^{'}$ make a couple. They can cross or not. It is defined by the crossover rate $cross_r$.

[1] We set as seven the limit of lessons to be permuted to allow the permutation to be computed in a viable processing time

Algorithm 1: Developed implementation of Memetic Algorithms

Input: Initial population $P = \{s_1, s_2, ..., s_n\}$.
Output: Best solution s^* found.
$s^* \leftarrow min(\{f(s_1), f(s_2), ..., f(s_n)\} \in P)$;
while $elapsedTime < timeout$ **do**
  $P \leftarrow Refinement(P)$;
  $P \leftarrow P \cup Crossover(P)$;
  $P \leftarrow Mutation(P)$;
  $P^{'} \leftarrow Selection(P)$;
  $P \leftarrow P^{'}$;
  $s^* \leftarrow min(\{f(s_1), f(s_2), ..., f(s_n)\} \in P)$;
return s^*;

Each pair of individuals (s, $s^{'}$) who breed, generate two children s_c and $s_c^{'}$, being s_c close to s and $s_c^{'}$ close to $s^{'}$. The children are originally just a copy of his closest parent. For each timeslot assignment in s (or $s^{'}$) we have a probability $swap_r$ to swap the timeslot assigned to the lesion at i-th position of s with the timeslot assigned at i-th position of $s^{'}$. This swap is made in s_c. The analogous swap is also made in $s_c^{'}$. Afterwards, the children s_c and $s_c^{'}$ are added to P. We empirically set $cross_r = 0.2$ and $swap_r = 0.01$.

2) Mutation: In the Mutation phase of MA, each individual $s \in P$ has a probability mut_r to be mutated. A mutation of an individual consists in the execution of a Lesson Swap or a Resource Swap movement in s. Both movements are selected with a 0.5 probability. After some experiments we set $mut_r = 0.1$.

3) Selection: To select the individuals who will survive to the next generation, we considered a tournament selection. Until n solutions be selected, we chose two solutions s_1 e s_2 and select the one who have the best fitness to survive to the next generation. We also implemented the elitism concept [26], since the best individual is always saved to the next generation.

4) Refinement: To refine the population, we apply the local search methods Simulated Annealing and Iterated Local Search to each individual in the population. The refinement method is almost the same developed by the ITC2012 winner [9], therefore, we set the parameters of both algorithms to consume less time.

5) Simulated Annealing: Proposed by [27], the metaheuristic Simulated Annealing is a probabilistic method based on an analogy to thermodynamics simulating the cooling of a set of heated atoms. This technique starts its search from any initial solution. The main procedure consists of a loop that randomly generates, at each iteration, one neighbor $s^{'}$ of the current solution s. Movements are probabilistically selected considering a temperature T and the cost variation of the movement Δ.

The developed implementation of Simulated Annealing is described in Algorithm 2. Parameters used were $\alpha = 0.97$, $T_0 = 1$ and $SAmax = 10.000$. The method $selectMovement()$ just chooses a movement according to the neighborhood probabilities previously defined.

Algorithm 2: Developed implementation of SA

Input: $f(.), N(.), \alpha, SAmax, T_0, s, timeout$
Output: Best solution s^* found.
$s^* \leftarrow s$; $IterT \leftarrow 0$; $T \leftarrow T_0$; $reheats \leftarrow 0$;
while $elapsedTime < timeout$ **do**
  while $IterT < SAmax$ **do**
    $IterT \leftarrow IterT + 1$;
    $k \leftarrow selectMovement()$;
    Generate a random neighbor $s^{'} \in N_k(s)$;
    $\Delta = f(s^{'}) - f(s))$;
    if $\Delta < 0$ **then**
      $s \leftarrow s^{'}$;
      if $f(s^{'}) < f(s^*)$ **then** $s^* \leftarrow s^{'}$;
    else
      Take $x \in [0, 1]$;
      if $x < e^{-\Delta/T}$ **then** $s \leftarrow s^{'}$;
  $T \leftarrow \alpha \times T$;
  $IterT \leftarrow 0$;
return s^*;

6) Iterated Local Search: The Iterated Local Search (ILS) method [28] is based on the idea that a local search procedure can achieve better results by optimizing different solutions generated through disturbances on the local optimum solution.

Our ILS algorithm starts from an initial solution s_0 obtained by the Simulated Annealing procedure and makes disturbances of size p_{size} under s_0 followed by a descent method. A disturbance is the unconditional acceptance of a neighbor generated by neighborhoods PR or KM, both with 0.5 of probability.

The descent phase just generates a neighbor and accepts him only if he is better than or match the current solution. Tested moves are excluded from the neighborhood and return only when an improvement to the current best solution is reached. The local search phase ends when there is no remaining neighbor to be explored.

The local search produces a solution $s^{'}$ which will be accepted if it is better than the best solution s^* found. In such case, the disturbance size p_{size} gets back to the initial size p_0. If the iteration $Iter$ reaches a limit $Iter_{max}$, the disturbance size is incremented. Yet, if the disturbance size reaches a bound p_{max}, it goes back to the initial size, p_0. The Algorithm 3 presents the developed implementation of ILS. The considered parameters were $ILS_{max} = 10,000$, $p_0 = 1$, $p_{max} = 10$ and $MaxIter_p = 10$.

IV. Computational Experiments

All experiments ran on an Intel ® i5 2.4 Ghz computer with 4GB of RAM running the Ubuntu 11.10 operating system. The programming language used on software development was C++ compiled by GCC 4.6.1. All of our results was validated by HSEval validator http://sydney.edu.au/engineering/it/~jeff/hseval.cgi. We considered a 1500 seconds timeout, adjusted according to the Third ITC provided benchmark, in all experiments.

Algorithm 3: Developed implementation of ILS

Input: $f(.), N(.), ILS_{max}, p_0, p_{max}, MaxIter_p, s, timeout$
Output: Best solution s^* found.
$s \leftarrow descentPhase(s)$; $s^* \leftarrow s$;
$p_{size} \leftarrow p_0$; $Iter \leftarrow 0$;
for $i \leftarrow 0$ *until* ILS_{max} **do**
  if $elapsedTime \leq timeout$ **then**
    for $j \leftarrow 0$ *until* p_{size} **do**
      $s \leftarrow s_p \in N(s)$;
    $s^{'} \leftarrow descentPhase(s)$;
    if $f(s^{'} < f(s^*))$ **then**
      $s \leftarrow s^{'}$; $s^* \leftarrow s^{'}$;
      $Iter \leftarrow 0$; $p_{size} \leftarrow p_0$;
    else
      $s \leftarrow s^*$;
      $Iter \leftarrow Iter + 1$;
    if $Iter = MaxIter$ **then**
      $p_{size} \leftarrow p_{size} + p_0$;
      if $p_{size} \geq p_{max}$ **then** $p_{size} \leftarrow p_0$;
return s^*;

The presented results are expressed by the pair x/y, where x contains the feasibility measure and y the quality measure. Our solver along with our solutions and reports can be found at https:// sites.google.com /site /georgehgfonseca /producaoacademica/ma.rar. We invite the interested reader to perform additional experiments and/or improve upon our existing solver.

A. Dataset Characterization

The set of instances available from Third ITC http://www.utwente.nl/ctit/ hstt/archives/XHSTT-2012 was originated from many countries and ranges from small instances to huge challenging ones. The Table I presents the main features of considered instances.

TABLE I. FEATURES OF CONSIDERED INSTANCES FROM THIRD ITC

Instance	Times	Teachers	Rooms	Classes	Lessons
BrazilInstance2	25	14		6	150
BrazilInstance3	25	16		8	200
BrazilInstance4	25	23		12	300
BrazilInstance6	25	30		14	350
FinlandElementarySchool	35	22	21	291	445
FinlandSecondarySchool2	40	22	21	469	566
Aigio1stHighSchool10-11	35	37		208	532
Italy_Instance4	36	61		38	1101
KosovaInstance1	62	101		63	1912
Kottenpark2003	38	75	41	18	1203
Kottenpark2005A	37	78	42	26	1272
Kottenpark2008	40	81	11	34	1118
Kottenpark2009	38	93	53	48	1301
Woodlands2009	42	40			1353
Spanishschool	35	66	4	21	439
WesternGreeceUniversity3	35	19		6	210
WesternGreeceUniversity4	35	19		12	262
WesternGreeceUniversity5	35	18		6	184

B. Obtained Results

The Table II presents the obtained results of Memetic Algorithms applied to the High School Timetabling Problem. We considered five executions of the algorithm to record the results, considering random seeds ranging from 1 to 5. The sub column $f(s^*)$ presents the cost of the best solution found, the sub column $f(\overline{s})$ presents the average cost of solutions and σ, the standard deviation.

TABLE II. OBTAINED RESULTS OF MEMETIC ALGORITHMS

Instance	Memetic Algorithms		
	$f(s^*)$	$f(\overline{s})$	σ
BrazilInstance2	0 / 45	0.0 / 64.4	0.0 / 19.0
BrazilInstance3	0 / 102	0.0 / 117.6	0.0 / 17.4
BrazilInstance4	1 / 147	1.4 / 157.6	0.9 / 10.2
BrazilInstance6	0 / 155	0.0 / 224.0	0.0 / 43.5
FinlandElementarySchool	0 / 4	0.0 / 4.0	0.0 / 0.0
FinlandSecondarySchool2	0 / 0	0.0 / 0.2	0.0 / 0.4
Aigio1stHighSchool10-11	0 / 6	0.0 / 13.6	0.0 / 6.5
Italy_Instance4	0 / 377	0.0 / 563.0	0.0 / 177.8
KosovaInstance1	324 / 19463	354.6 / 20883.4	22.0 / 2252.5
Kottenpark2003	2 / 75408	2.0 / 75408.0	0.0 / 0.0
Kottenpark2005A	38 / 31316	38.0 / 31316.0	0.0 / 0.0
Kottenpark2008	72 / 189885	72.0 / 189885.0	0.0 / 0.0
Kottenpark2009	26 / 16755	26.0 / 26886.0	0.0 / 13164.7
Woodlands2009	2 / 12	2.0 / 15.2	0.0 / 1.8
Spanish school	0 / 655	0.0 / 795.4	0.0 / 96.1
WesternGreeceUniversity3	0 / 5	0.0 / 5.8	0.0 / 0.4
WesternGreeceUniversity4	0 / 9	0.0 / 10.0	0.0 / 0.7
WesternGreeceUniversity5	0 / 0	0.0 / 0.0	0.0 / 0.0

We also reproduced the results of Third ITC winner, a standalone SA-ILS approach. A comparison of these results and the MA approach are presented in the Table III. In the column SA-ILS [9] we have the average results of the ITC2012 winner and in the column MA we have the results of the proposed method. The last row presents the ranking of the two solvers considering the ITC2012 ordering procedure. Briefly speaking, each solver receives a rank to each instance ranging from 1 (best) to 2 (worst) according to the average cost of solutions obtained. The solver with the smaller average is considered the best.

TABLE III. COMPARSION BETWEEN SA-ILS APPROACH AND MA APPROACH

Instance	SA-ILS [9]	MA
BrazilInstance2	1.0 / 63.9	**0.0 / 64.4**
BrazilInstance3	0.0 / 127.8	**0.0 / 117.6**
BrazilInstance4	17.2 / 99.6	**1.4 / 157.6**
BrazilInstance6	4.0 / 223.5	**0.0 / 224.0**
FinlandElementarySchool	0.0 / 4.0	0.0 / 4.0
FinlandSecondarySchool2	0.0 / 0.4	**0.0 / 0.2**
Aigio1stHighSchool10-11	0.0 / 15.3	**0.0 / 13.6**
Italy_Instance4	0,0 / 658.4	**0.0 / 563.0**
KosovaInstance1	**14.0 / 6934.4**	354.6 / 20883.4
Kottenpark2003	**0.6 / 90195.8**	2.0 / 75408.0
Kottenpark2005A	**33.9 / 27480.4**	38.0 / 31316.0
Kottenpark2008	**25.7 / 31403.7**	72.0 / 189885.0
Kottenpark2009	36.6 / 154998.5	**26.0 / 26886.0**
Woodlands2009	2.0 / 15.8	**2.0 / 15.2**
Spanish school	0.0 / 865.2	**0.0 / 795.4**
WesternGreeceUniversity3	**0.0 / 5.6**	0.0 / 5.8
WesternGreeceUniversity4	**0.0 / 7.4**	0,0 / 10.0
WesternGreeceUniversity5	0.0 / 0.0	0.0 / 0.0
Ranking	1.61	**1.39**

C. Discussion of Results

For some instances, even the production of feasible solutions configures a hard task. These instances commonly define most of constraints as hard constraints. The Memetic Algorithms approach was able to find 11 out of 18 feasible solutions to the considered instance set, the same amount that the Third ITC winner found.

As Table II show, the Memetic Algorithms were able to overcome the Third ITC winner. It reached a ranking of 1.39, far smaller than 1.61, ranking achieved by the SA-ILS approach. Looking at tables I and III, we can note that the MA approach achieved the best results to 9 out of 11 [2] instances with 1.000 or less lessons to schedule. However, to the instances with more than 1.000 lessons to schedule, the SA-ILS approach got better results, beating the MA approach to 4 out of 7 instances. Thus, we can conclude that the new approach is more suitable to handle small instances of the problem.

To explain why we got better results to the smaller instances, we can take a look at the Memetic Algorithms procedure. It runs the local search (SA-ILS) several times (6 per generation). To the large instances, we generally cannot run the local search this amount of times since we ran out of time. Several times, it happens without even a single generation to be fully processed. In other hand, to the small instances, we can run the local search phase several times. Moreover, the local search is made in several solutions, which lead the algorithm to explore better the solution space. This fact may explain the superior performance of MA algorithms over the standalone SA-ILS approach.

V. Concluding Remarks

The Memetic Algorithms with a Refinement phase composed by a SA-ILS approach showed strong results applied to the High School Timetabling Problem. It even overcomes the Third ITC winner approach, a standalone SA-ILS algorithm. This result shows that evolutionary approaches can be applied to improve the performance of local search methods through the Memetic Algorithms. The MA allows the local search to take advantage of the evolutionary algorithms features.

Some possible future works are (1) implement and evaluate other evolutionary strategies to this problem; (2) implement other neighborhood movements; and (3) run more experiments, with a larger timeout to evaluate better the MA performance applied to the larger instances of the problem.

Acknowledgment

The authors acknowledge FAPEMIG (grants APQ-01779-10) and CNPq (grant 480388/2010-5) for supporting the development of this research.

References

[1] M. R. Garey and D. S. Jonhson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, USA, 1979.

[2] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM Jounal of Computing*, vol. 5, no. 4, pp. 691–703, Dec. 1976.

[3] IDSIA, "International Timetabling Competition 2002," 2012, available at http://www.idsia.ch/Files/ttcomp2002/, Accessed in December / 2012.

[4] P. Kostuch, *The university course timetabling problem with a three-phase approach*, ser. Proceedings of the 5th international conference on Practice and Theory of Automated Timetabling. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 109–125.

[5] S. Kirkpatrick, D. C. Gellat, and M. P. Vecchi, *Otimization by Simulated Annealing*, ser. Science, 1983, pp. 202, 671–680.

[2]Note that to 2 of them the results were equal.

[6] B. McCollum, "International Timetabling Competition 2007," 2012, available at http://www.cs.qub.ac.uk/itc2007/, Accessed in December / 2012.

[7] T. Muller, *ITC2007 solver description: a hybrid approach.*, ser. Annals OR, 2009, vol. 172, no. 1, pp. 429–446.

[8] U. of Twente, "International Timetabling Competition 2012," 2012, available at http://www.utwente.nl/ctit/hstt/itc2011/welcome/, Accessed in December / 2012.

[9] G. Fonseca, H. Santos, T. Toffolo, S. Brito, and M. Souza, *A SA-ILS approach for the High School Timetabling Problem*, ser. PATAT '12 Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling, 2012.

[10] H. G. Santos, E. Uchoa, L. S. Ochi, and N. Maculan, *Strong bounds with cut and column generation for class-teacher timetabling*, ser. Annals OR, 2012, vol. 194, no. 1, pp. 399–412.

[11] G. Post, J. Kingston, S. Ahmadi, S. Daskalaki, C. Gogos, J. Kyngas, C. Nurmi, N. Musliu, N. Pillay, H. Santos, and A. Schaerf, "XHSTT: an XML archive for high school timetabling problems in different countries," *Annals of Operations Research*, pp. 1–7.

[12] J. H. Kingston, *A tiling algorithm for high school timetabling*, ser. Lecture notes in computer science: V Practice and theory of automated timetabling. Berlin: Springer, 2005, pp. 3616 : 208–225.

[13] M. Wright, *School timetabling using heuristic search*, ser. Journal of Operational Research Society, 1996, pp. 47 : 347–357.

[14] K. Nurmi and J. Kyngas, *A framework for school timetabling problem*, ser. Proceedings of the 3rd multidisciplinary international scheduling conference: theory and applications, Paris, 2007, pp. 386–393.

[15] C. Valourix and E. Housos, *Constraint programming approach for school timetabling*, ser. Computers & Operations Research, 2003, pp. 30 : 1555–1572.

[16] P. de Haan, R. Landman, G. Post, and H. Ruizenaar, *A case study for timetabling in a Dutch secondary school*, ser. Lecture notes in computer science: VI Practice and theory of automated timetabling. Berlin : Springer, 2007, pp. 3867 : 267–279.

[17] G. Post, S. Ahmadi, S. Daskalaki, J. H. Kingston, J. Kyngas, C. Nurmi, and D. Ranson, *An XML format for benchmarks in High School Timetabling*, ser. Annals of Operations Research DOI 10.1007/s10479-010-0699-9., 2010, pp. 3867 : 267–279.

[18] J. H. Kingston, "A software library for school timetabling," 2012, available at http://sydney.edu.au/engineering/it/~jeff/khe/, May 2012.

[19] ——, "Hierarchical timetable construction," in *Proceedings of the First International Conference on the Practice and Theory of Automated Timetabling*, 2006.

[20] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, Sep. 2003. [Online]. Available: http://dx.doi.org/10.1145/937503.937505

[21] A. R. R. Freitas, F. G. Guimares, R. C. Pedrosa Silva, and M. Souza, "Memetic self-adaptive evolution strategies applied to the maximum diversity problem," *Optimization Letters*, pp. 1–10, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11590-013-0610-0

[22] A. S. Ruela, R. da Silva Cabral, A. L. L. Aquino, and F. G. Guimares, "Memetic and evolutionary design of wireless sensor networks based on complex network characteristics," *IJNCR*, vol. 1, no. 2, pp. 33–53, 2010.

[23] T. Liu, Z. Jiang, and N. Geng, "A memetic algorithm with iterated local search for the capacitated arc routing problem," *International Journal of Production Research*, no. ahead-of-print, pp. 1–10, 2013.

[24] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[25] P. Moscato, "New ideas in optimization," D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, and K. V. Price, Eds. Maidenhead, UK, England: McGraw-Hill Ltd., UK, 1999, ch. Memetic algorithms: a short introduction, pp. 219–234. [Online]. Available: http://dl.acm.org/citation.cfm?id=329055.329078

[26] B. Chakraborty and P. Chaudhuri, "On the Use of Genetic Algorithm with Elitism in Robust and Nonparametric Multivariate Analysis," *Austrian Journal of Statistics*, vol. 32, no. 1, pp. 13–27, 2003. [Online].

Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.58.472

[27] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[28] H. R. Lourenco, O. C. Martin, and T. Stutzle, "Iterated local search," in *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Boston: Kluwer Academic Publishers, 2003, ch. 11.