A Simulated Annealing Algorithm with a new Neighborhood Structure for the Timetabling Problem

Yongkai Liu
Department of Computer Science, Xiamen University
Xiamen, 361005, China
+86
yongkailiu@sina.com

Defu Zhang
Department of Computer Science, Xiamen University
Xiamen, 361005, China
+86 592 5918207
dfzhang@xmu.edu.cn

Stephen C.H. Leung
Department of Management Sciences, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong
mssleung@cityu.edu.hk

ABSTRACT
In this paper, a new neighborhood structure is presented. The new neighborhood is obtained by performing a sequence of swaps between two timeslots, instead of only one move in the standard neighborhood structure. Based on new neighborhood, simulated annealing algorithm can solve the timetabling problem well. The computation results on two open benchmarks coming from two real-world high schools timetabling problems prove that the simulated annealing algorithm based on new neighborhood can compete with other effective approaches.

Categories and Subject Descriptors
F.2.2

General Terms
Algorithms

Keywords
Timetabling; Simulated Annealing; Neighborhood.

1. INTRODUCTION
The complexity [1] of the school timetabling problems is increasing due to the large enrollment of students every year. It is more and more difficult to produce high-quality timetables now. However, over the last few years, several meta-heuristics, such as Ant Colony Optimization, Evolution Algorithm, Tabu Search and Simulated Annealing (SA) , have been successfully applied to the timetabling problems and the packing problems, and some good results were reported [2~13].

SA has been proved to be an effective algorithm in solving NP-complete problem [11,12]. During these years, it has been frequently developed and employed to solve timetabling problems. In 1991, it was firstly applied to solve high school timetabling problems by D. Abramson [13]. Later, it was improved by Abramson et al [7] to produce school timetables. In their works, they designed special dynamic cooling schedule for SA, in which the temperature can adaptively reduce or increase according to the number of success moves. In 2001, SA was modified by Burke et al [6] to solve exam timetabling problems. The algorithm is able to improve the timetable gradually according to the user's predefined time. Recently, Pasquale Avella et al [8] used SA to construct a feasible timetable in the first phase of their algorithm, while Philipp Kostuch [9] employed SA to optimize the timetable in the third phase of his work. In this paper, SA with a new neighborhood structure is developed and applied to two typical real-world high school timetabling problems: the Hdtt case (HCT) in [8] and the Greek case (GCT) in [3].

GEC'09, June 12-14, 2009, Shanghai, China.

2. PROBLEM DEFINITION
There are various definitions [14] for the timetabling problem. A popular one is that timetabling problem deals with allocating a set of events (lessons, exams) to some limited resources, such as rooms and timeslots (hours), subject to a series of constraints. These constraints are often grouped into hard constraints and soft constraints. Hard constraints can not be violated under any circumstances, while soft constraints should be satisfied as much as possible.

In high school, students are often organized into class to have their lessons in fixed classroom, while teachers are specified to teach one or more lessons on several classes. For each week, every class has to finish the specified number of lessons, while every teacher has to teach the specified number of lessons on the corresponding class. HCT and GCT are defined as allocating all the lessons to timeslots, subject to a series of constraints. The constraints of HCT and GCT are similar, and they are summarized as follows:

(H1) No teacher can teach two or more lessons in the same timeslot.
(H2) No class can attend two or more lessons in the same timeslot.
(H3) Every teacher must teach the exact number of lessons specified.
(H4) Every class must attend the exact number of lessons specified.
(H5) Every teacher must work in his/her available timeslots.
(H6) The classing hours of each class must be continuous during the day.
(H7) The non-classing hour is only allowed to occur in the last timeslot of the day.

(S1) The teaching hours of each teacher should be continuous during the day.
(S2) Teacher's teaching hours should be balanced throughout the week.
(S3) Each class should not attend the same lesson twice or more on one day.
(S4) Teacher should not be allocated to the timeslots she/he is unwilling to work in.

Rules (H1)-(H7) are hard constraints, and they are used to determinate the feasibility of a timetable. Rules (S1)-(S4) are soft constraints, and they are used to evaluate the quality of a timetable. The violations of hard constraints are called hard lashes while the violations of soft constraints are called soft clashes. We only accept a feasible timetable without hard clashes, and use some criterions concerned with soft clashes to evaluate the feasible timetable. However, the evaluation methods of HCT and GCT are different, so they are adopted respectively when we make comparisons.

In HCT, let $\boldsymbol{T}$ be the set of teachers, $\boldsymbol{C}$ be the set of classes, $\boldsymbol{S}$ be the set of timeslots, $\boldsymbol{D}$ be the set of days, β_t be the punishment coefficients of total soft clashes of teacher t, and a_i be the punishment coefficients of criterion i (see below). The objective function of HCT [8] is defined as follows:

$$\sum_{t\in T}\beta_t(a_1\sum_{c\in C}\sum_{s\in S}\gamma_{tcs}X_{tcs}+a_2\sum_{s\in S}Z_{ts}+a_3\sum_{c\in C}\sum_{s\in S}Y_{tcs}$$
$$+a_4\sum_{c\in C}\sum_{d\in D}F_{tcd}+a_5\sum_{d\in D}(G_{td}+H_{td})+a_6\sum_{c\in C}T_{tc})$$

The criterions and the notation of variables in this function are defined as follows:

1. The penalty of allocating teacher to the timeslot which the teacher is unwilling to work in is calculated by the first item of the function, where X_{tcs} =1 indicates teacher t has a lesson on class c in timeslot s (otherwise X_{tcs} =0), and γ_{tcs} is the punishment coefficient of teacher t teaches class c in timeslot s.
2. The penalty of teacher's idle hours is calculated by the second item of the function, where Z_{ts} =1 indicate teacher t has an idle hour in timeslot s, otherwise Z_{ts} =0.
3. The penalty of teacher's idle hours on the specified class is calculated by the third item of the function, where Y_{tcs} =1 indicates teacher t has an idle hour on class c in timeslot s, otherwise Y_{tcs} =0.
4. The penalty of teachers not teaching average hours on the specified class is calculated by the fourth item of the function, where F_{tcd} =1 indicates teacher t does not teach average hours on class c on day d, otherwise F_{tcd} =0.
5. The penalty of teachers not having balance workload through a week is calculated by the fifth item of the function, where G_{td} =1 indicates teacher t teaches less than the specified minimal hours on day d (otherwise G_{td} =0), while H_{td} =1 indicates teacher t teaches more than the specified maximal hours on day d (otherwise H_{td} =0).
6. The penalty of teacher working in the last timeslot of the day is calculated by the sixth item of the function, where T_{tc} =1 indicates teacher t has a lesson on class c in the last timeslot of the day, otherwise T_{tc} =0.

In GCT, the quality of the timetable is evaluated by six criterions. They are denoted as: *DT, TDT, DL, TDL, TG* and *TTG*. *DT* is the number of teachers whose teaching hours has uneven distribution on some day, while *TDT* records the total number of such days. *DL* is the number of classes which attend the same lessons more than once on some day, while *TDL* records the total number of such days. *TG* is the number of teachers whose teaching hours is not continuous on some day, while *TTG* records the total number of idle hours for all teachers. More details can be found in [3].

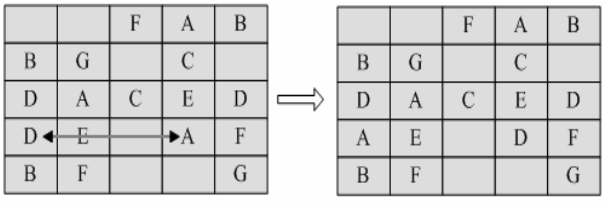
3. THE NEIGHBORHOOD STRUCTURE

The solution is represented by a two-dimensional matrix $\boldsymbol{C}\times\boldsymbol{S}$, where the row represents class and the column represents timeslot. If there is a t allocated in row c and column s, it indicates that teacher t has a lesson on class c in timeslot s. A feasible solution should include all the teacher-class pairs (lessons). Figure 1 shows a simple example of a solution, where *A, B, C, D, E, F,* and *G* are the allocated teachers.

	timeslot 1	timeslot 2	timeslot 3	timeslot 4	timeslot 5
class 1			F	A	B
class 2	B	G		C	
class 3	D	A	C	E	D
class 4	D	E		A	F
class 5	B	F			G

Figure 1. An example of solution

Generally, a standard neighborhood [8, 13] is defined as all the solutions reachable from the current one by performing only one move, such as swapping two teachers and swapping a teacher and an empty position in the same row between two timeslots. Just as showed in Figure 2, by swapping the teacher *A* and *D* in the fourth row, we obtain a new solution (b) of the neighborhood, where the hard clashes in the first timeslot are reduced, and the idle hours of teacher *D* are also reduced.



(a) **(b)**

Figure 2. An example about the standard neighborhood

Note that there are still some other potential improving moves between these two timeslots of Solution (b), so it may be promising if we continue to search between these two timeslots. In addition, after swapping two teachers between these two timeslots, the clashes of both teachers are probably reduced, so we believe that it can be efficient to concentrate on searching between these two timeslots.

Inspired by this idea we developed a new neighborhood structure, in which a neighborhood is obtained by performing a sequence of swaps between two timeslots. The swaps in the sequence are performed one by one, but whether a swap can be accepted is decided by the SA approach, in which the improving move and the side move (which keeps the same value of objective function) are always accepted, while the deteriorating move is accepted with a probability:

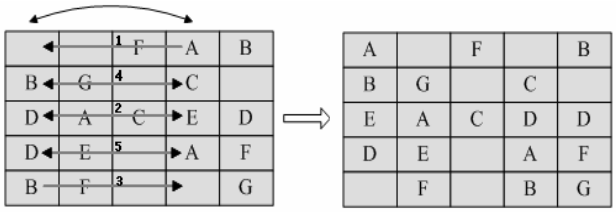
$$P(\Omega_c \leftarrow \Omega_n \mid T, \Delta) = e^{-\frac{\Delta}{T}}$$

Where Ω_n is the new solution obtained by performing a move on the current one Ω_c, Δ is the value of the deterioration of the objective function, T is the current temperature, and it reduces or increases according to the cooling schedule. The sequence is generated by randomly sorting all the possible swaps between two timeslots. The reason we try to perform all the possible swaps is that we believe it can keep the diversity and flexibility of the timetable by accepting more side moves, so as to avoid dropping into local optima too soon.

We give an example to describe how to generate and perform a sequence of swaps between two timeslots in Figure 3. Let ~ denote an empty position of the matrix, and $\leftrightarrow$ denote swapping two teachers or swapping a teacher and an empty position in the same row. In this example, between the first timeslot and the fourth timeslot, there are five swaps: $(\sim \leftrightarrow A)$, $(B \leftrightarrow C)$, $(D \leftrightarrow E)$, $(D \leftrightarrow A)$ and $(B \leftrightarrow \sim)$. By randomly sorting these five swaps, we suppose the sequence of swaps we obtain is:

$$\xrightarrow{1} (\sim \leftrightarrow A) \xrightarrow{2} (D \leftrightarrow E) \xrightarrow{3} (B \leftrightarrow \sim) \xrightarrow{4} (B \leftrightarrow C) \xrightarrow{5} (D \leftrightarrow A)$$

Considering the hard clashes, when this sequence of swaps is performed, swap 1, swap 2 and swap 3 are improving moves and they are performed one by one, but swap 4 and swap 5 become deteriorating moves after the first three swaps are performed, so they are accepted with a probability. One possible solution obtained by performing these swaps is shown as Figure 3(b). One can easily find that there are no hard clashes in the new solution (b).



(a) **(b)**

Figure 3. An example of the neighborhood

4. ALGORITHM DESCRIPTION

There are two phases in our algorithm (SA3). The first phase aims to construct a feasible solution. The second phase concentrates the effort on improving the quality of the timetable while maintaining feasibility. The Pseudo-code of the algorithm is showed in Fig 4.

Phase 1: Construct a feasible timetable

When initializing the solution, we only consider the hard constraint (H3) and (H4), and allocate all the teachers to the matrix. For each teacher, according to the number of teaching hours specified on the class, the teacher is allocated to the same number of positions in the corresponding row. One position of the matrix can be only occupied by one teacher.

We adopt a stochastic strategy in this phase. At each inner loop of SA, we randomly select a timeslot where the hard clashes happen, and randomly select another one (no matter whether it violates hard constraints or not) from the rest timeslots. Then we generate a sequence of swaps between these two selected timeslots and try to perform every swap in this sequence one by one. The experiment showed that this approach rarely fails in producing a feasible solution, and its average performance is higher than many heuristic construction algorithms. In Figure 4, we present the pseudo-code of creating a feasible timetable in phase 1.

A greedy strategy, in which selecting two worst timeslots with most hard clashes was also designed and tested. The experiment showed that it was easy to get stuck in the local optima in the later improving phase. One possible explanation is that the search space is limited due to only selecting some of the worst timeslots repeatedly.

Phase 1: Create a feasible timetable

Initialize: allocate all the teachers to the solution matrix

// $H(\Omega)$ returns the number of hard clashes of solution Ω

$Hardclashes \leftarrow H(\Omega_c)$

while($Hardclashes$ >0 **and** T >terminal temperature)

for $k \leftarrow 0$ **to** K // K is the number of inner loops

select a timeslot i with hard clashes

select another one j randomly

generate a sequence of swaps between i and j

for each swap m in the sequence

$\Omega_n \leftarrow$ perform m on Ω_c

$newHardclashes \leftarrow H(\Omega_n)$

$\Delta \leftarrow newHardclashes - Hardclashes$

// $Rand()$ returns a random real number between 0 and 1

if($\Delta \le 0$ **or** $e^{-\frac{\Delta}{T}} > Rand()$)

$Hardclashes \leftarrow newHardclashes$

$\Omega_c \leftarrow \Omega_n$

$T \leftarrow \alpha * T$ // α is the temperature cooling rate

Go to Phase 2

Figure 4. Pseudo-code of Phase 1

Phase 2: Optimize the feasible solution

Once a feasible timetable is found, we turn to reduce the soft clashes. It is not allowed to bring hard clashes in this phase.

We adopt a tabu search strategy in this phase. A tabu list is used to lock the selected timeslots and a candidate list is used to contain the rest timeslots. At each inner loop, the tabu list is cleared, and all the timeslots are randomly sorted and added into the candidate

list. At each iteration of inner loop, the first timeslot of candidate list is selected and coupled with one of the other in the candidate list, and then we generate a sequence of swaps between these two coupled timeslots and try to perform all the swaps in this sequence one by one while maintaining the feasibility of the timetable. After this iteration, the first timeslot is removed from candidate list and added into the tabu list. When there is only one timeslot left in the candidate list, a new inner loop starts. The pseudo-code of phase 2 is shown in Figure 5.

Phase 2: Optimize the feasible solution

// $S(\Omega)$ returns the number of soft clashes of solution Ω

$Softclashes \leftarrow S(\Omega_c)$

while(T > terminal temperature)

  for $k \leftarrow 0$ **to** K // K is the number of inner loops

    add all the timeslots in candidate list

    clear the tabu list

    for each of the timeslot $i \neq j$ in candidate list

      generate a sequence of swaps between i and j

      for each swap m in the sequence

        $\Omega_n \leftarrow$ perform m on Ω_c

        if($H(\Omega_n) = 0$)

          $newSoftclashes \leftarrow S(\Omega_n)$

          $\Delta \leftarrow newSoftclashes - Softclashes$

          if($\Delta \leq 0$ **or** $e^{-\frac{\Delta}{T}} > Rand()$)

            $Softclashes \leftarrow newSoftclashes$

            $\Omega_c \leftarrow \Omega_n$

    remove j from candidate list and add it into the tabu list

  $T \leftarrow \alpha * T$ // α is the temperature cooling rate

Output: return the final solution

Figure 5. Pseudo-code of Phase 2

An example of the search process is shown in Figure 6. We suppose the timeslots in candidate list are ordered as: i, j, and k. Timeslot i is firstly selected and coupled with timeslot j, and then coupled with k. Just as the example shown in section 3, after performing the sequence of swaps between i and j on Solution (a), we obtain a new Solution (b), and then after performing the sequence of swaps between i and k on Solution (b), we obtain another new Solution (c). One can easily find that idle hours occur in timeslot i is reduced in the new solution (c).

5. COMPUTATIONAL RESULTS

5.1 The benchmarks

Two benchmarks [15, 16] were chosen to test our algorithm, and their instance details are shown in Table 1 and Table 2. The first benchmark was contributed by Dr. Kate Smith [17]. This benchmark was also tested by Pasquale Avella et al [8] and Smith KA et al [17]. The second benchmark was derived from real-word Greek high-school timetabling problems, and it contains 11 instances. The first seven instances were collected by Grigorios et al [3], while the last four instances were collected by K Papoutsis et al [18]. We didn't test the high_school_06 because we still don't have access to all the details of this instance.

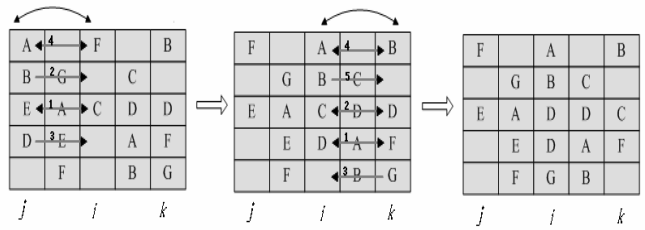


Figure 6. An example of the search process

The problem size in the first benchmark is small in terms of the number of classes and teachers. But the timetable is dense from the perspective of assignments, because there are no empty positions left after all teachers are initially allocated into the matrix. The problems in the second benchmark have more lessons, classes, teachers, and timeslots. They are more complicated and difficult than the first benchmark.

The algorithms were implemented in c++, and they are carried out on a personal computer with Pentium® 4 CPU 2.60 GHz and 512M RAM.

5.2 The computational results of HCP

The empirical parameters of SA are showed in Table 3. As in [8], we set the coefficients β_s=1, a_i=1 and γ_{tcs}=0, that is, there are no teacher preferences and every criterion has the same importance. As in [8], the best results, average results and average time were obtained by 20 runs of the program. They are compared with SA [8], NN-TT3 [17] and SA2 [17] in Table 4.

The results show that our algorithm SA3 is able to return high-quality timetable without designing special cooling schedule, and it provides the best solution among the algorithms so far. It is easy to get to a conclusion that SA with our new neighborhood structure indeed has higher efficiency and performance than SA with the other two neighborhood structures.

Table 1. Instance details of HCP

Instance	04	05	06	07	08
Teachers	4	5	6	7	8
Classes	4	5	6	7	8
Timeslots	30	30	30	30	30

Table 2. Instance details of GCP

Instance	01	02	03	04	05	06	07	08	09	10	11
Teachers	34	35	19	19	18	34	35	11	15	17	21
Classes	11	11	6	7	6	10	13	5	6	7	9
Timeslots	35	35	35	35	35	35	35	30	35	30	35

Table 3. Parameters of the SA3 (HCP)

Initial temperature (phase1)	Initial temperature (phase2)	Terminal temperature (phase1)	Terminal temperature (phase2)
4.0	1.0	0.1	0.05
α (phase1)	α (phase2)	Number of inner loop(phase1)	Number of inner loop(phase2)
0.90	0.90	20	2

Table 4. Computational results of HCP

		SA	NN-TT3	SA2	SA3
hdtt 04	Best	0	0	0	0
	Average	0	0.5	0	0
	Time(sec)	0.63	1.9.2	14.57	0.203
hdtt 05	Best	0	0	0	0
	Average	0	0.5	0.3	0
	Time(sec)	1.54	146.3	41.2	0.578
hdtt 06	Best	0	0	0	0
	Average	0	0.7	0.8	0
	Time(sec)	3.47	226.9	123.02	3.652
hdtt 07	Best	0	0	0	0
	Average	0.1	1	1.2	0
	Time(sec)	7.29	323.7	256.59	5.359
hdtt 08	Best	0	0	0	0
	Average	0.6	1.2	1.9	0.4
	Time(sec)	14.68	431.3	471.2	9.140

5.3 The computational results of GCP

Setting appropriate coefficients for each criterion of the objective function is very significant. In [3], the EA is designed as a one-phase algorithm, and the weight of each criterion varies according to some variables, such as the idle hours of the teachers and dispersion of the lessons. However, coefficients are set to be constant in our algorithm and the empirical values are shown in Table 5.

Table 5. Coefficients of criterion of GCP

Test Data	DT	TDT	DL	TDL	TG	TTG
01-07	3	5	20	20	1	4
08-11	7	8	9	9	12	12

The empirical parameters of SA are shown in Table 6. We ran ten times of our program (however, this number was not mentioned in [3]), and recorded the best results to compare with the EA algorithm in Table 7. The results show that our algorithm has a superior advantage in constructing a feasible timetable, and it is able to produce better timetable in much shorter time. The experiment demonstrates that the two-phase algorithm is a very promising approach, and it also proves that the approaches based on the single solution improving strategy (such as SA) are often more efficient than the approaches based on the population solutions improving strategy (such as EA) in solving timetabling problems.

Table 6. Parameters of SA3 (GCP)

Initial temperature (phase1)	Initial temperature (phase2)	Terminal temperature (phase1)	Terminal temperature (phase2)
1.0	1.0	0.1	0.05
α (phase1)	α (phase2)	Number of inner loop(phase1)	Number of inner loop(phase2)
0.95	0.90	100	10

6. Conclusions and Further Work

We have proposed a new neighborhood structure in this work, and it has been successfully combined with SA to solve two typical high-school timetabling problems. The computational results show that our algorithms can compete with other effective approaches, and also prove that the new neighborhood structure is able to increase the efficiency and performance of SA. Though the sequence of swaps is constituted by horizontal moves in one row, it can be modified for other search process (such as an example shown in Figure 7), in which swapping across rows is also allowed. Further, it can be easily developed to cooperate with other algorithms. Our next work is to combine this neighborhood structure with other meta-heuristics to solve large-scale timetabling problems.

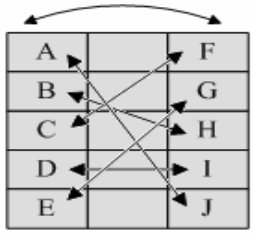


Figure 7. An example of another neighborhood operator

7. ACKNOWLEDGMENTS

Special thanks go to Charalampos N. Moschopoulos and Grigorios N.Beligiannis for providing more details about the Greek case test data sets and their valuable suggestion. We would also like to thank K. Papoutsis for his test data and his advice. This work was supported by the National Nature Science Foundation of China (Grant no. 60773126) and the Province Nature Science Foundation of Fujian (Grant no. A0710023)

Table 7. Computational results (GCP)

	EA					SA3				
TestData	DT(TDT)	DL(TDL)	TG(TTG)	Time (min)		DT(TDT)	DL(TDL)	TG(TTG)	Phase 1 Time(sec)	Phase 2 time(sec)
01	18(40)	1(1)	25(29)	30		6(11)	1(1)	11(17)	1.234	138.625
02	15(34)	0(0)	26(42)	30		7(14)	3(3)	10(20)	1.078	170.212
03	4(8)	3(3)	9(9)	24		3(4)	3(3)	1(1)	0.468	47.140
04	5(12)	0(0)	17(29)	24		4(11)	0(0)	7(17)	0.593	62.656
05	1(2)	0(0)	8(8)	22		0(0)	0(0)	3(3)	0.562	53.609
07	24(50)	13(27)	24(32)	45		7(12)	3(4)	11(24)	6.484	202.609
08	3(6)	5(15)	0(0)	22		3(5)	5(15)	0(0)	0.203	20.453
09	7(14)	6(21)	0(0)	24		5(11)	6(17)	0(0)	0.375	30.578
10	2(4)	7(22)	0(0)	24		2(6)	7(17)	0(0)	0.234	32.156
11	6(13)	9(29)	0(0)	28		5(11)	9(21)	0(0)	0.656	79.390

8. REFERENCES

[1] T.B.Cooper, J.H.Kingston. 1995. The complexity of timetabling construction problems. Proceedings of the 1st International Conference on Practice and Theory of Automated Timetabling LNCS 1153, Berlin: Springer, 283-295.

[2] Krzysztof Socha, Joshua Knowles, Michael Sampels. 2002. A MAX-MIN Ant System for the University Course Timetabling Problem. Lecture notes in computer science, vol 2463, Berlin: Springer, 1-13.

[3] Grigorios N. Beligiannis, Charalampos N. Moschopoulos, Georgios P. Kaperonis, Spiridon D.Likothanassia. 2008. Applying evolutionary computation to the school timetabling problem: The Greek case. Computer & Operations Research, 35(4): 1265-1280.

[4] Di Gaspero, L., Schaerf. 2001. A. Tabu search techniques for examination timetabling. Selected Papers from the 3rd International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, vol. 2079, 104 – 117.

[5] Halvard Arntzen, Halvard Arntzen, Arne Løkketangen. 2003. A tabu search heuristic for a university timetabling problem. The Fifth Metaheuristics Internetional Conference.

[6] E. Burke, Y. Bykov, J. Newall, S. Petrovic. 2001. A Time-Predefined Local Search Approach to Exam Timetabling Problems. Computer Science Technical Report No. NOTTCS-TR-2001-6, Univ. of Nottingham.

[7] Abramson, D., Krishnamoorthy, M., Dang, H. 1999. Simulated annealing cooling schedules for the school timetabling problem. Asia-Pacific Journal of Operational Research 16, 1-22.

[8] Pasquale Avella, Bernardo D'Auria Saverio Salerno, Igor Vasil'ev. 2007. A computational study of local search algorithms for Italian high-school timetabling. Springer Science+Business Media, LLC.

[9] Philipp Kostuch. 2004. The University Course Timetabling Problem with a Three-Phase Approach. Proceedings of the 5th International Conference on Practice and Theory of Automated Timetabling V LNCS 1153, Berlin: Springer, 109-125.

[10] ZHANG De-Fu, CHEN Sheng-Da, LIU Yan-Juan. 2007. An improved heuristic recursive strategy for genetic algorithm of the orthogonal packing of rectangles ACTA AUTOMATICA SINICA 33(9):911-916.

[11] ZHANG De-Fu, Li Xin. 2005. A Personified Annealing Algorithm for Circles Packing Problem ACTA AUTOMATICA SINICA. 31(4):590-595.

[12] Lijun Wei, Defu Zhang,Qingshan Chen. 2008. A least wasted first heuristic algorithm for rectangular packing problem Computers & Operations Research 2008.In press, http://dx.doi.org/10.1016/j.cor.

[13] D. Abramson. 1991. Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms. Management Science, vol. 37, no.1, 98-113.

[14] Burke, E.K., Jackson, K.S., Kingston, J.H, Weare, R.F. 1997. Automated timetabling: The state of art. Comput. J. 40(9), 565-571

[15] http://prlab.ceid.upatras.gr/timetabling/.

[16] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/tableinfo.html.

[17] Smith KA, Abramson D, Duke D. 2003. Hopfield neural networks for timetabling: formulations, methods, and comparative results. Computers and Industrial Engineering 44(2): 283–305.

[18] K Papoutsis, C Valouxis and E Housos. 2003. A column generation approach for the timetabling problem of Greek high schools, Journal of the Operational Research Society, 54: 230–238.