New Trends in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics Volume II: Applications

Editors

Krassimir T. Atanassov
Władysław Homenda
Olgierd Hryniewicz
Janusz Kacprzyk
Maciej Krawczak
Zbigniew Nahorski
Eulalia Szmidt
Sławomir Zadrożny

SRI PAS  IBS PAN

New Trends in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics
Volume II: Applications

Editors

Krassimir Atanassov
Władysław Homenda
Olgierd Hryniewicz
Janusz Kacprzyk
Maciej Krawczak
Zbigniew Nahorski
Eulalia Szmidt
Sławomir Zadrożny

IBS PAN iBS PAN **SRI PAS**

Systems Research Institute
Polish Academy of Sciences
Newelska 6, 01-447 Warsaw, Poland
www.ibspan.waw.pl

ISBN 83-894-7547-2

Proposed multi–criterion optimisation method of school timetabling problem

Jacek M. Czerniak, Wojciech T. Dobrosielski
Casimir the Great University, Institute of Technology,
ul. Chodkiewicza 30, 85-064 Bydgoszcz, Poland
jczerniak@ukw.edu.pl, wdobrosielski@ukw.edu.pl
and
Rafal A. Angryk
Montana State University Department of Computer Science
Bozeman, MT 59717–3880,USA
angryk@cs.montana.edu

Abstract

Authors of the article propose multi–criterion optimisation method PRTS dedicated for optimizing software–generated university timetables. In their research they used real data made available by Casimir the Great University in Bydgoszcz. PRTS method is based on tabu search mechanisms. It means that this is a metaheuristic that searches for the problem solution by supervising other heuristic procedures of local searching, in order to explore the space of solutions beyond the local optimum. The solution space searching process is coordinated by means of strategies based on memory mechanisms which are characteristic features of TS algorithm.

Keywords: tabu search, timetabling, scheduling, course timetabling, meta–heuristics.

1 Introduction

The professional literature [1, 2, 4] dedicated to multi–criterion optimisation includes diverse and interesting approaches to problems called timetabling and scheduling. On the other hand, authors many times faced nuisance of non–optimum

New Trends in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics. Volume II: Applications *(K.T. Atanassow, W. Homenda, O. Hryniewicz, J. Kacprzyk, M. Krawczak, Z. Nahorski, E. Szmidt, S. Zadrożny, Eds.), IBS PAN - SRI PAS, Warsaw, 2013.*

planning consequences. This paper is only one of the diverse approaches to the problem of creating optimum university timetables using metaheuristics. In this very case the PRTS method was developed based on tabu search mechanisms. Timetabling consists in establishing lecturer–group combinations for specified periods of time and groups of students. As the optimising quality criterion, authors assumed reduction of conflicts occurring when several lecturers [3, 7] must give lectures or classes for the same group of students in the same time periods or when different lecturers or classes require the same room. The main trouble is associated with the scale of the problem [19, 20]. On one hand, lecturers' needs (who expect specific timetable arrangement) should be taken into account and on the other hand students studying hygiene should be taken into consideration. Those factors should be considered in the context of education law as well as internal regulations of university authorities that require, for example, minimized number of free periods. Very often those aims are mutually exclusive and the only solution is to find the best compromise [8].

2 Tabu search algorithm

TS is a metaheuristic that searches for the problem solution by supervising other heuristic procedures of local searching, in order to explore the space of solutions beyond the local optimum [5, 6]. The process of searching the space of solutions is coordinated by means of strategies based on memory mechanisms which are characteristic features of TS algorithm [19].

TS algorithm performs full search of the neighbourhood of the current solution [9, 10, 11]. Current solution is replaced by the best solution in the neighbourhood, even if it deteriorates the quality. The searching process uses the system of limitations imposed on the collection belonging to the neighbourhood. The objective is to prevent the possibility of the algorithm looping and returning to the same narrow solution spaces. Solutions that were previously accepted as current solutions are removed from the defined neighbourhood thus creating so called tabu collection [18]. Limitations used in TS can have a form of absolute ban or only some restrictions.

Algorithm 1 The pseudo code presenting the operation of Tabu Search algorithm

Algorithm start

Initialisation of the memory structures
Selection of the initial solution x and determining its quality $f(x)$

while $f(x) = \sum w_r f_r(x) \geq \Theta$ **do**

- Determining the best solution found so far $x* := x$
- Generating neighbourhood $N(x)$ of the current solution
- Selection of the best neighbourhood $N*(x)$ taking into account tabu rules generated on the basis of data stored in the memory structures
- Selection of the best solution x'
- The best solution becomes the current solution $x := x'$

if $x' > x*$ **then**
$x* := x'$
end if
end while

Algorithm stop

The main and most characteristic mechanism of tabu search is the memory. Information on the searched space is collected there during searching process [12, 13]. Local selections depend on information collected during the entire searching process. Information collected in the memory is basis for creation of limitations which protect the algorithm against returning to already searched areas of the solution space [14, 16]. Those limitations depend on the following factors:

- Recording frequencies for specified data,
- validity of data stored in the memory,
- influence of specific data on quality of achieved results.

In consecutive iterations, the algorithm searches the neighbourhood of the currently found solution in order to determine new location of the current solution. Therefore it is necessary to define a neighbourhood relation for all items of the set search space.

The memory structure first and foremost stores information on executed moves. It can also include information e.g. on frequency of moves or the time

that has elapsed from the execution of each move. There are two types of memory in TS algorithms: long-term and short-term one. Each memory type is used by its characteristic strategies. The result of their operation can be observed as modification of neighbourhood $N(x)$ of the current solution x. Modified neighbourhood $N*(x)$ is the result of stored information concerning searching process completed so far.

Short-term memory is used during every iteration to store most recently visited solutions and penalties imposed on them. Its main task is to avoid selection of the move operator which could lead to algorithm looping in some narrow area of the solution space.

Long-term memory stores information on the course of the search process. It allows to store the best solutions from already visited areas of the solutions space, instead of the best solution from the current neighbourhood only [15, 17]. To determine the neighbourhood relation, it is necessary to specify the operator for the change of the current solution which defines the neighbourhood of the point: any point that can be obtained from the current point using the change operator is a neighbour of that current point unlike other points which are not its neighbours.

A collection of moves to new locations belonging to the neighbourhood of the current solution must be specified for each step of the algorithm. One of them shall be selected as the new current solution. It is necessary to specify and check all attributes of any generated move. If the value of such attribute is defined by at least one tabu limitation, then such attribute is considered as tabu-active. Otherwise the attribute is considered tabu-inactive. Once the status of all attributes for each new transition is specified, the rules specifying if a given move is tabu-active or tabu-inactive are generated. Tabu-active status characterising one move attribute does not necessarily mean that status of the entire move is also tabu-active. This depends on the rules which use attribute statuses. For example, if a given rule specifies that status of the move to a new location is a Boolean sum of two specific attributes, then tabu-inactiveness of both those attributes makes the entire transition tabu-inactive. To set-up the status of a given attribute as tabu-active it is always necessary to specify the number of iterations during which that status shall be maintained. Once this number of iterations has been executed, the attribute status is changed into tabu-inactive. However, if a tabu move is made before, which activates this attribute again, then the activeness period for that attribute is counted from the beginning. The length pf the tabu period does not have to be uniform for all attributes, but is should rather be individually specified for each of them. This length may either be constant or variable over time. In that case, the length may be fixed (i.e. it would be the function of the number of currently performed iteration) or may change in a random way. It may also be

the function of factors defining the progress made by the algorithm. Every move may be defined by different attributes. This may for example include the value of the penalty assigned to it, number or recent frequency of this move, the tabu period, i.e. the number of remaining iterations for which the ban on that move remain in force. All such information shall be stored in the memory structures during algorithm execution. Those two event types differ, i.e.: high incidence of certain value at certain position means that this value of attribute is particularly desirable. Whereas high variability of the value on a specified position indicates to high unpredictability of that attribute and it means that its role is probably higher in the final tuning to optimum than in the initial optimization phase, i.e. in the initial exploration aimed at finding the most promising area of the domain.

New value of the current solution is searched for as follows: the operator of the move to the next position selects some subset of solutions from the neighbourhood of the current position and searches for the best one among them. However, it determines values and availability of solutions taking into account tabu rules. Once the set of neighbours is generated, it is analysed and, after all the tabu rules are taken into account, the best individual found in that set becomes the new current solution of the algorithm. Information about that move is stored in the memory structure.

The aim of tabu rules, used by the move operator, is to prevent the search process getting stuck in local optimums. They are also used to withdraw from already visited areas of the solution space, which enables the algorithm to search wider space. There are diverse tabu rules. They may introduce total or partial bans. Those rules may also take into account the time. For example, the ban on move between points may be cancelled after predefined period of time (i.e. number of iterations) since the last such move. Penalties may also be diminished and finally cancelled after some time. However, they are resumed when such move is made again.

Tabu limitations may prohibit attractive moves, even when there is no risk of getting stuck in local optimums or they may lead to stagnation of the searching process. So it is necessary to use tools which will make possible to cancel the tabu. They are called aspiration criteria. Limitations may be softened by cancelling a selected limitation and adding some penalty for the infringement of the prohibition. There are several ways to determine penalty values. One of them is the use of so called self-adjusting penalty. Its value changes dynamically based on the search process history.

3 Implementation of ts-aided timetabling

The timetabling process was implemented in the Matlab environment and it consists in generating a timetable for each lecturer and each group of students. The schedule of use for individual rooms is also created. The set of all partial timetables forms the resulting timetable. The aim of the programme is to reduce conflicts occurring when several lecturers must give lectures or classes for the same group of students in the same time periods or when different lecturers or classes require the same room. The main trouble is associated with the scale of the problem. Apart from a large number of groups, lecturers, classes and rooms, numerous additional conditions must also be taken into account.

There are two types of limitations in timetabling. Heavy limitations (H) must be absolutely met. Whereas conditions considered as useful but not necessary in a good timetable are called soft limitations (S). In the process of searching for optimum solution, the value of this parameter is directly proportional to the achieved quality of the timetable. Therefore they influence the value of the evaluation function.

The timetable consists of the set of m lecturers $t_1, \ldots, t_m$ giving lectures and classes to n groups of students $c_1, \ldots, c_n$ during the time periods $p_1, \ldots, p_j$. Assignment of lecturers to individual groups of students is pre-defined, whereas their workload is defined in the matrix of requirements. The problem is solved by minimization the value of evaluation function $f(x)$ which takes into account the degree to which limitations are met by generated solutions.

The evaluation function is denominated by the formula:

$$f(x) = \sum w_r f_r(x) \tag{1}$$

where r is the consecutive limitation number.

Weight values w_r are defined by the developer and they specify the importance of a given limitation in the entire process. Functions $f_r(x)$ are summed up values of all solutions x_{ita} for parameters defined by a given limitation. Subtotals associated with heavy limitations define the feasibility level, whereas subtotals connected to soft limitations are the measure of satisfaction.

There are following heavy limitations assumed at the stage of the programme creation:

(H1) Each lecture or class must be assigned to a specific time slot or its multiple, depending on the duration of an individual class.

(H2) None of the lecturers can give two different classes at the same time.

(H3) None of the groups can take part in two different classes at the same time.

(H4) Two concurrent classes cannot be given in the same room.

(H5) A class or a lecture must take place in a room of appropriate type.

(H6) Availability/unavailability periods for individual teachers must be adhered to

(H7) Classes or lectures of individual student groups of the same year class cannot collide with common classes or lectures for that year class.

Soft limitations take the following form:

(S1) Number of free periods in timetables of individual student groups should be reduced to minimum in order to eliminate idle time.

(S2) Number of free periods in timetables of individual teachers should be reduced to minimum in order to eliminate idle time.

(S3) The number of classes of the same type (subject/teacher/class type) students have on the same day should not exceed the specified value.

The maximum number of classes (limitation S3) is defined by the user. It is also possible to change the maximum number of iterations, the number of meetings for specified schedule and the type of studies. In case of extramural studies classes are held from Friday to Sunday and the user can select the time when the first class starts on the first day of the meeting. Classes and lectures of the full-time course are held on weekdays.

Fixed parameters of the algorithm include:

- tabu time,
- weights w_r of the evaluation function,
- number of iterations followed by intensification/diversification.

The timetabling (generating the timetable) must be preceded by input of basic information. They include data referring to:

- lecturers,
- class years,
- groups,

- subjects,
- rooms,
- types of classes (lecture, lab etc.).

This data is used to create a schedule, i.e. the list of all classes to be held in a given semester/year. The timetable is generated based on information included in the schedule.

The abovementioned data are stored in form of tables linked to each other by specific relations, presented in fig.(1). The assumed time unit amounts to 15

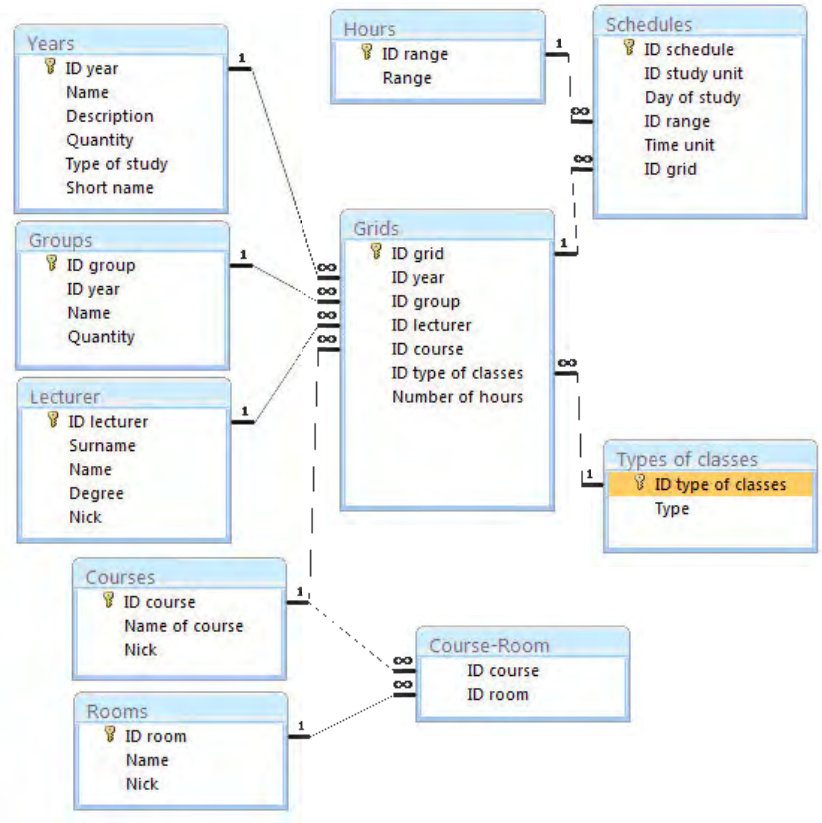

Figure 1: Data structure and mutual relations between them

minutes.

Discussed implementation used three three-dimensional matrixes: T, C and S, to store individual partial timetables. First of them includes information on

lecturers' timetables t, the second one about timetables of student groups c, and the third one −about rooms s. Those tables constitute separate structures, but they

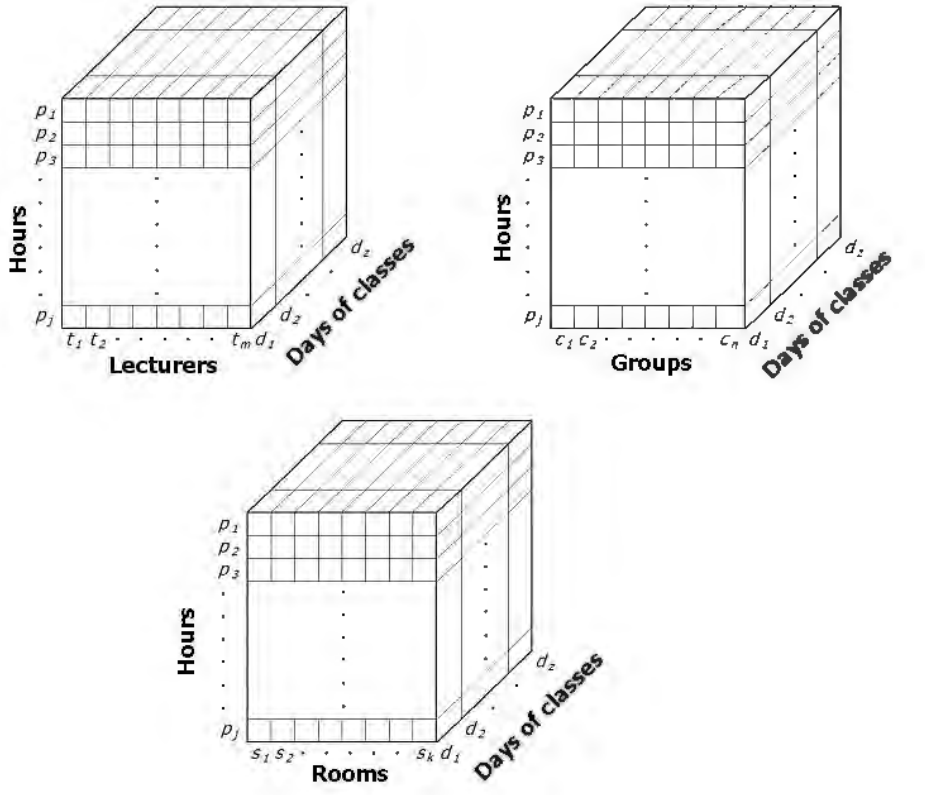

Figure 2: Matrixes storing partial timetables for: a) lecturers, b) groups, c) rooms

are linked to the schedule table. If randomly selected item of the schedule table (id schedule) is placed in one of the matrixes T, C or S, then that item is also placed in analogous positions of the other two matrixes. Due to the fact that each item of id schedule explicitly defines the lecturer, the group and the room it refers to, only one value is stored in each cell of the matrixes T, C and S. Otherwise it would be necessary to place more information in matrix cells. Fig. (3) and (4) illustrate the manner in which information is stored in matrixes T, C and S and the Schedule table.

The purpose of the programme is to generate the timetable with the least possible value of the evaluation function. While determining this value, the degree to which the function is satisfied by the generated result of limitations specified in point 3.2 is also analysed. Implemented algorithm prevents creation of the

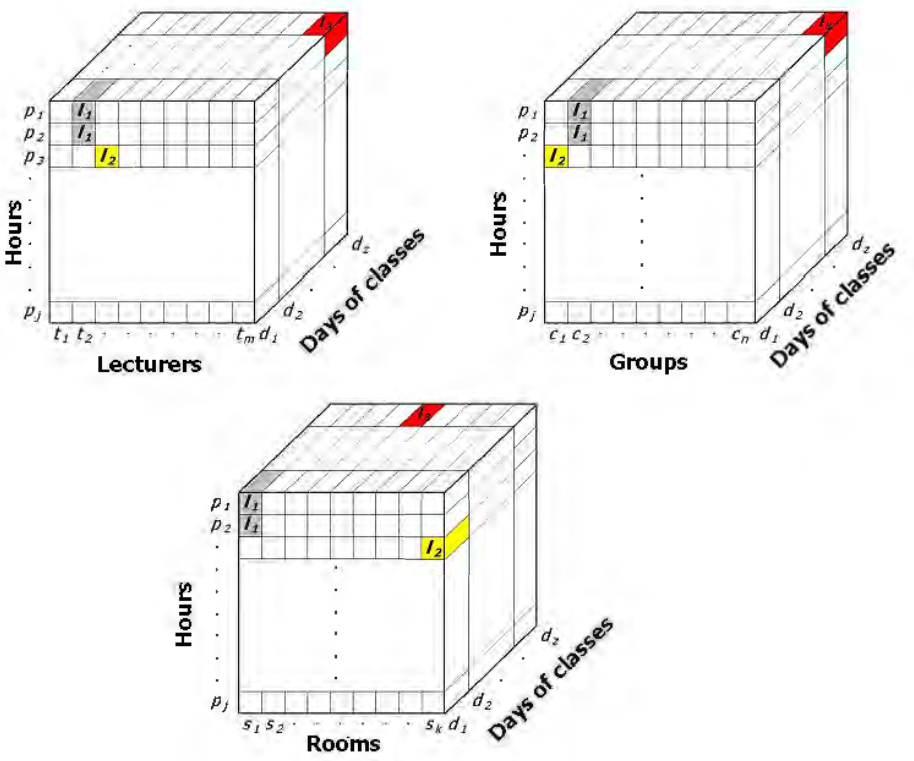

Figure 3: Way of data storage in matrixes T, C and S. Item l_i represents i–th value of id schedule

timetable breaching any of heavy limitations (H1)–(H7). Thus, only soft limitations (S1)–(S3) are taken into account to determine the value of the evaluation function. A penalty point is imposed for each breaching of the limitations. The value of the evaluation function of the generated timetable is equal to the sum of all penalty points of all partial timetables.

The timetable generation process starts from randomly generated initial solution. The space of solutions X is searched during each iteration. This space is the set of solutions satisfying the limitations (H1)–(H7).

It would demand lots of calculations to examine the entire neighbourhood of the solution x. For that reason the programme uses the strategy of the list of potential items. Each individual class–lecturer–group–room combination (id schedule) is selected randomly, but the randomness is limited. The solution x' must meet heavy limitations, so it is not necessary to search the entire neighbourhood $N(x)$.

The tabu list and attributes of accepted moves are stored in the programme

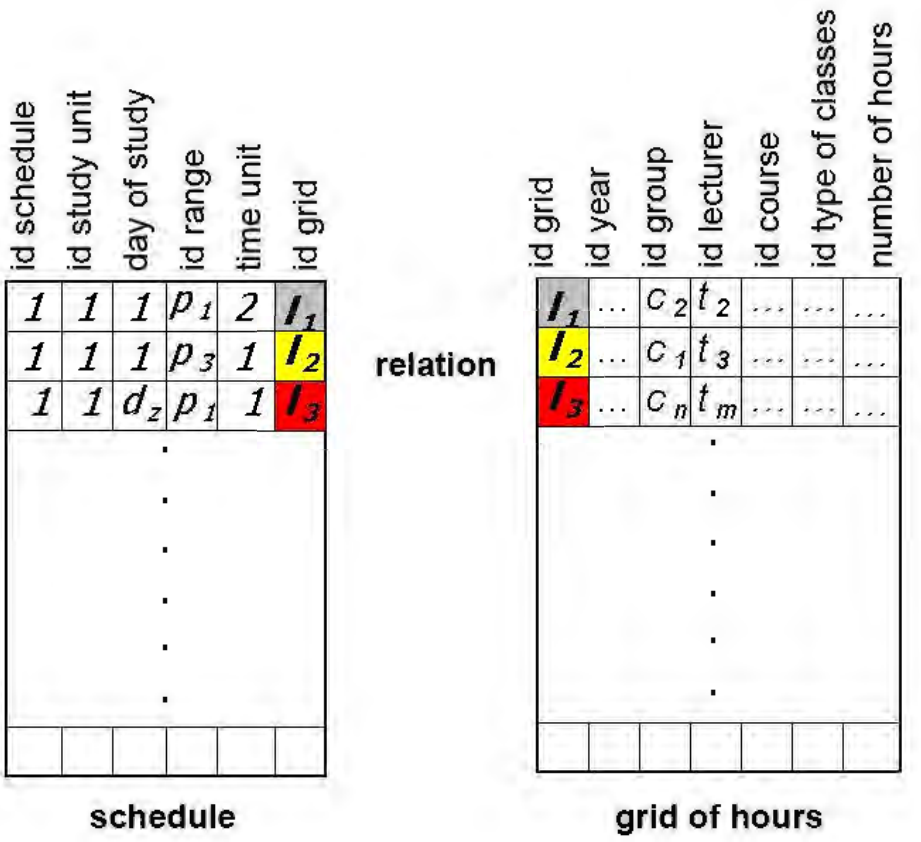

Figure 4: The manner in which individual items of the generated timetable are stored based on data from matrixes T, C and S in fig.(3)

memory. Number of repetitions for which the move shall be prohibited is also stored there. This number is decreased during each consecutive iteration and once it reached zero value the move is removed from the list. Whereas the move selected to be executed generates solution x' with the best value of the evaluation function, which is better than the evaluation function value of the previously achieved best solution.

Once the move is made, the solution characterized by the best value of the evaluation function in the new region is stored in the memory. If better solution than the stored one is not reached after defined number of repetitions then the stored solution shall be examined once more (intensification). Thus the searching process focuses on neighbours of good solutions. If no solution which is better than the best local solution is found within defined number of iterations, then the

algorithm returns to the best local solution.

All algorithm steps described above are executed until one of the termination criteria is met: i.e. the solution characterised by the evaluation function value equal 0 is found or number of iterations reaches the set point.

4 Results

The experiment was divided in two parts. First of them included measurement of the time needed for random generation of the timetable without optimizing algorithms [20].

The efficiency of the method and the quality of achieved timetable were examined. Efficiency was measured by execution of the application 100 times and then by determining the number of correctly generated timetables. The timetabling time and the quality of generated timetables was determined based on 50 correctly selected timetables in a random way. Whereas in the second part of the experiment analogous measurements were performed for timetables generated using TS algorithm. The tabu period equal 5 was assumed without the possibility of conditional execution of prohibited move. Currently searched neighbourhood was abandoned in favour of new areas after 10 iterations. Weight values wr of the evaluation function was fixed at 0.01, both for partial values of evaluation function associated with timetables of individual lecturers and individual student groups. Thanks to introduction of equal weight values, each factor is equally important in the evaluation of the timetable quality. Maximum number of iterations amounted to 500. The timetable was generated for full-time course.

Results of the experiment discussed in point 3.6 are illustrated in fig.(5–6)

Time necessary for random generation of a timetable is shorter for a random process (fig.(5)). The first generated solution becomes the final one regardless of the value of the evaluation function, which is confirmed in the fig. (6 a, b). On average, the quality of the resulting timetable obtained using TS algorithm is twice as good as in case of random timetabling. The value of the evaluation function is decisive for the final form of the resulting timetable. The differences shown as graphs in the fig. (7 a, b) are clearly visible in the graphic version of timetables.

5 Conclusions

The results presented above prove that use of PRTS algorithm significantly increases the quality of the final solution compared to results of random generation. The quality of local searching is also improved. The limitation system makes it

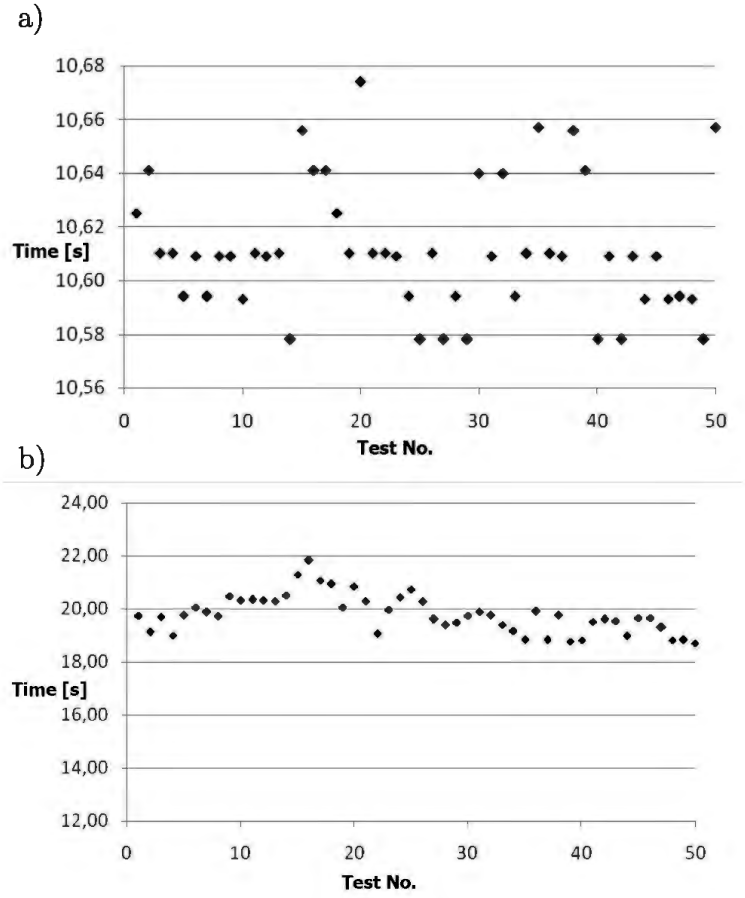

Figure 5: Time necessary to generate a timetable: a) randomly, b) using TS

possible to examine all available versions of the timetable being developed and to determine the version that best meets the predefined assumptions. It is somewhat difficult to compare the results with results achieved by other authors, as virtually all of them used their own data which were often stored in unique formats. Thus it is very difficult to use them. But this is usually not the major obstacle. In most cases this data is sensitive in respect of personal data protection or the interest of the organisation it belongs to. Authors of this study would like to thank the management of The Institute of Technology at Casimir the Great University in Bydgoszcz for providing access to data used for experiments presented herein.

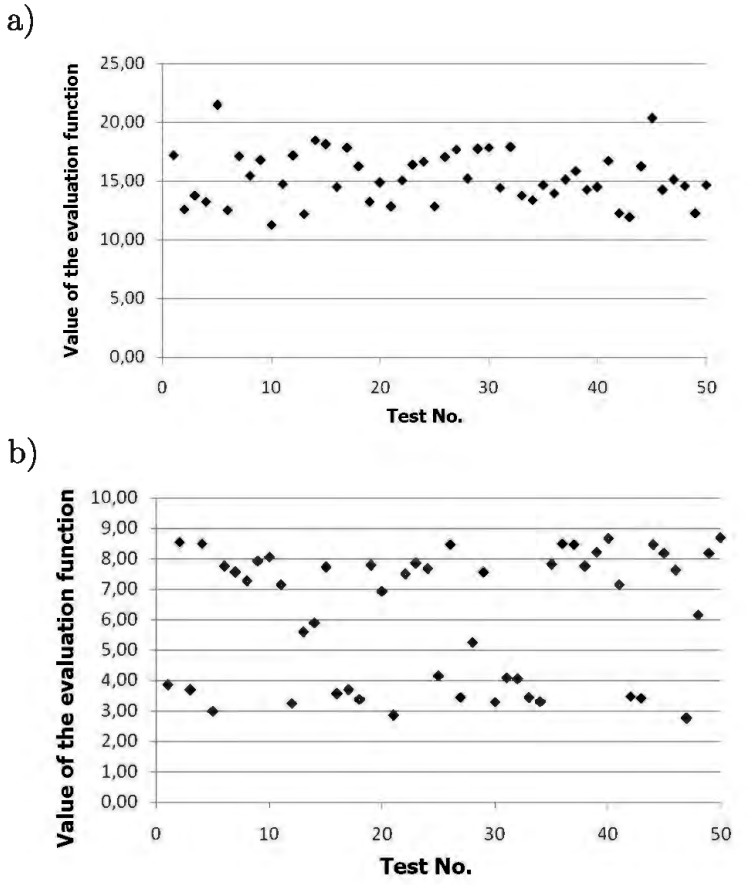

Figure 6: Quality of a generated timetable: a) randomly, b) using TS

References

[1] Aladag, .H., Hocaoglu, G., 2007, A Tabu Search Algorithm to Solve a Course Timetabling Problem, Hacettepe Journal of Mathematics and Statistics, vol. 36(1), s.53-64

[2] Aladag, C.H., Hocaoglu, G., 2009, Basaran, M.A.,The effect of neighborhood structures on tabu search algorithm in solving course timetabling problem Expert Systems With Applications Volume: 36, Issue: 10, December, pp. 349–356

[3] Alvarez–Valdes, R., Crespo, E., Tamarit, J.M., 2002. Design and implemen–

a)

Hours	Monday	Tuesday	Wednesday	Thursday	Friday
07:00 - 08:00				N. t. i.,J. C., ól,15	
08:00 - 09:00		D. t., i. i p. z.,E. K., ól,1			
09:00 - 10:00	Inż. wytw. III,T. W., ól,7				P. i. i s. i.,I. F., ól,17
10:00 - 11:00	Elektr.,A. M., w,2	Sem. mag.,J. C., s,26	Sem. mag.,T. W., s,26	B. d.,C. G., ól,1	N. o m. III,J. P., w,14
11:00 - 12:00				P. k. i e. m. I,K. T., ól,18	
12:00 - 13:00	P. i. i s. i.,I. F., w,17			S. k. i a. s ,M. D., ól,21	N. t. i.,J. C., w,15
13:00 - 14:00	P. k. i e. m. II,K. T., ól,19	Pedag.,A. K., k,16		Inż. wytw. III,T. W., w,7	Pr. mag.,T. W., s,22
14:00 - 15:00	P. k. i e. m. II,K. T., w,19		Mech. tech.,J. K., ól,10	Mech. tech.,K. C., k,10	D. t., i. i p. z.,M. K.,A., w,1
15:00 - 16:00				Sem. mag.,J. K., s,26	
16:00 - 17:00	N. o m. III,J. P., ól,14		Pr. mag.,J. K., s,22	Mech. tech.,K. C., w,10	Pedag.,A. K., w,16
17:00 - 18:00		P. k. i e. m. I,K. T., w,18			K. w. w t.,K. W., ól,9
18:00 - 19:00			Pr. mag.,J. C., s,22	D. t., i. i p. z.,E. K., k,1	S. k. i a. s .,J. C., w,27
19:00 - 20:00	Elektr.,A. M., ól,2	P. k. i e. m. II,K. T., k,19	K. w. w t.,M. M., w,9		
20:00 - 21:00			Pr. k. II,R. S., ól,21	B. d.,C. G., w,1	

b)

Hours	Monday	Tuesday	Wednesday	Thursday	Friday
07:00 - 08:00					
08:00 - 09:00	N. t. i.,J. C., ól,15	Inż. wytw. III,T. W., ól,7		N. t. i.,J. C., w,15	D. t., i. i p. z.,E. K., ól,1
09:00 - 10:00	N. o m. III,J. P., ól,14	Mech. tech.,J. K., ól,10		K. w. w t.,K. W., ól,9	P. i. i s. i.,I. F., ól,17
10:00 - 11:00	D. t., i. i p. z.,M. K.,A., w,1			P. k. i e. m. I,K. T., ól,18	Sem. mag.,J. K., s,26
11:00 - 12:00	N. o m. III,J. P., w,14	Pr. mag.,T. W., s,22			Elektr.,A. M., ól,2
12:00 - 13:00	Pr. k. II,R. S., ól,21	Inż. wytw. III,T. W., w,7	P. k. i e. m. II,K. T., k,19	Pr. mag.,J. K., s,22	Elektr.,A. M., w,2
13:00 - 14:00	S. k. i a. s .,M. D., ól,21	Sem. mag.,J. C., s,26	P. k. i e. m. II,K. T., ól,19	B. d.,C. G., ól,1	
14:00 - 15:00	B. d.,C. G., w,1	P. k. i e. m. I,K. T., w,18	P. k. i e. m. II,K. T., w,19	Pedag.,A. K., w,16	
15:00 - 16:00	Mech. tech.,K. C., w,10	P. i. i s. i.,I. F., w,17	Mech. tech.,K. C., k,10		
16:00 - 17:00		Sem. mag.,T. W., s,26	K. w. w t.,M. M., w,9		
17:00 - 18:00		D. t., i. i p. z.,E. K., k,1	Pedag.,A. K., k,16		
18:00 - 19:00			S. k. i a. s .,J. C., w,27		
19:00 - 20:00			Pr. mag.,J. C., s,22		
20:00 - 21:00					

Figure 7: Sample timetables generated: a) randomly, b) using TS

tation of a course scheduling system using tabu search. European Journal of Operational Research 137 (3), 512–523

[4] Burke, E.K., Petrovic, S., 2002. Recent research directions in automated timetabling. European Journal of Operational Research, vol. 140 (2), pp.266–280

[5] Burke, E.K., Kingston, J., Jackson, K., Weare, R., 1997. Automated university timetabling: The state of the art. The Computer Journal, vol. 40 (9), 565–571

[6] Caldeira, J.P., Agostinho, C.R., 1997. School timetabling using genetic search. In: Proceedings of the Practice and Theory of Automated Timetabling, University of Toronto, Toronto, pp.115–122

[7] Costa, D., 1994. A tabu search algorithm for computing an operational timetable. European Journal of Operational Research, vol. 76 (1), 98–110

[8] Daskalaki, S., Birbas, T., Housos, E., 2004. An integer programming formulation for a case study in university timetabling. European Journal of Operational Research, vol. 153 (1), 117–135

[9] Erben, W., Keppler, J., 1996. A genetic algorithm solving a weekly course-timetabling problem. In: Burke, E., Ross, P. (Eds.), Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, vol. 1153. Springer, Berlin, pp. 198–211

[10] Gaspero, L., Schaerf, A., 2006, Neighborhood Portfolio Approach for Local Search Applied to Timetabling Problems Journal of Mathematical Modelling and Algorithms Volume: 5, Issue: 1, April, pp. 65 −89

[11] Gaspero, L.D., Schaerf, A., 2001. A case-study for Easy-Local++: The course timetabling problem. Research Report, Universita degli Studi di Udine, Italy

[12] Glover, F., 1986, Future Paths for Integer Programming and Links to Artificial Intelligence, Computer and Operations Research, vol. 13, no. 5, ss. 533–549.

[13] Glover, F., 1989, "Tabu Search −Part I," INFORMS Journal on Computing, vol. 1, no. 3, pp. 190–206

[14] Glover, F., 1990, "Tabu Search −Part II," INFORMS Journal on Computing, vol. 2, no. 1, pp. 4–32

[15] Hansen, P., 1986, The steepest ascent mildest descent heuristic for combinatorial programming, Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy,

[16] L, Z., Hao, J.K., , 2010, Adaptive Tabu Search for course timetabling, European Journal of Operational Research Volume: 200, Issue: 1, January pp. 235–244

[17] Pongcharoen, P., Promtet, W., Yenradee, P., Hicks, C., 2008, Stochastic Optimisation Timetabling Tool for university course scheduling International Journal of Production Economics, Volume: 112, Issue: 2, April, pp. 903–918

[18] Schuster, Ch.J., 2006,No–wait Job Shop Scheduling: Tabu Search and Complexity of Subproblems Mathematical Methods of Operations Research, Volume: 63, Issue: 3, July , pp. 473 –491

[19] Socha, K., Sampels, M., Manfrin, M., 2003. Ant algorithms for the university course timetabling problem with regard to the state–of–the–art. In: Proceedings of the Third European Workshop on Evolutionary Computation in Combinatorial Optimisation. Springer, UK.

[20] Souza, M.J.F., Maculan, N., Ochi, L.S., 2001. A GRASP–tabu search algorithm to solve a school timetabling problem. In: Proceedings of the Fourth Metaheuristics International Conference, Porto, Portugal, pp. 53–58

The papers presented in this Volume 2 constitute a collection of contributions, both of a foundational and applied type, by both well-known experts and young researchers in various fields of broadly perceived intelligent systems.
It may be viewed as a result of fruitful discussions held during the Eleventh International Workshop on Intuitionistic Fuzzy Sets and Generalized Nets (IWIFSGN-2012) organized in Warsaw on October 12, 2012 by the Systems Research Institute, Polish Academy of Sciences, in Warsaw, Poland, Institute of Biophysics and Biomedical Engineering, Bulgarian Academy of Sciences in Sofia, Bulgaria, and WIT - Warsaw School of Information Technology in Warsaw, Poland, and co-organized by: the Matej Bel University, Banska Bystrica, Slovakia, Universidad Publica de Navarra, Pamplona, Spain, Universidade de Tras-Os-Montes e Alto Douro, Vila Real, Portugal, Prof. Asen Zlatarov University, Burgas, Bulgaria, and the University of Westminster, Harrow, UK:

Http://www.ibspan.waw.pl/ifs2012

The Workshop has also been in part technically supported by COST Action IC0806 ``Intelligent Monitoring, Control and Security of Critical Infrastructure Systems'' (INTELLICIS).

The consecutive International Workshops on Intuitionistic Fuzzy Sets and Generalized Nets (IWIFSGNs) have been meant to provide a forum for the presentation of new results and for scientific discussion on new developments in foundations and applications of intuitionistic fuzzy sets and generalized nets pioneered by Professor Krassimir T. Atanassov. Other topics related to broadly perceived representation and processing of uncertain and imprecise information and intelligent systems have also been included. The Eleventh International Workshop on Intuitionistic Fuzzy Sets and Generalized Nets (IWIFSGN-2012) is a continuation of this undertaking, and provides many new ideas and results in the areas concerned.

We hope that a collection of main contributions presented at the Workshop, completed with many papers by leading experts who have not been able to participate, will provide a source of much needed information on recent trends in the topics considered.

ISBN-13 9788389475473