A Multi-Neighborhood Multi-Operator Algorithm for the Uncapacitated Exam Proximity Problem

Tony WONG, Salah ASSAL
Automated Manufacturing Engineering, École de technologie supérieure
Montréal, H3C 1K3, Canada

ABSTRACT

An algorithm featuring multiple local search operators and multiple neighborhood structures is applied to the uncapacitated exam proximity problem. The use of multiplicity is to enable effective interplay between intensification and diversification during the search process. The algorithmic design is inspired by Hansen and Mladenovic's Variable Neighborhood Descent algorithm and the "one operator, one landscape" point of view. Its performance was evaluated using publicly available datasets. For the uncapacitated exam proximity problem, the multi-neighborhood and multi-operator algorithm compared favorably against other search techniques. These results should encourage further research on the application of multiple operator approach as solution techniques to the uncapacitated exam proximity problem.

Keywords: exam proximity problem, uncapacitated, multiple neighborhood, multiple operator, local search.

1. INTRODUCTION

The Uncapacitated Exam Proximity Problem (UEPP) is a simplified variant of the general Exam Timetabling Problem. In the UEPP, the main objective is to provide to the students as much free time as possible between successive exams without taking into account the total sitting capacity. Usually, there exists a set of requirements that determine the timetable characteristics. For example, a timetable must avoid assigning students to more than one exam per timeslot. Also, it is desirable for a timetable to have prescribed length. Thus, the timetable construction is a process that maximizes the temporal separation of the exams while satisfying the given constraints.

In this work, an algorithm is conceived by integrating several local search operators to explore and exploit the search space. Each local search operator is associated with a different neighborhood structure so that the search effort is enhanced. The result is a multi-operator search algorithm that is simple in terms of implementation and effective in terms of solution quality.

This paper is organized as follows. Section 2 defines the UEPP using graph-theoretic approach. Section 3 presents the datasets used in the benchmarking and a brief survey on other solution methods. The survey is restricted to previously published methods related to the datasets provided by Carter et al. [1], Burke et al. [2] and Merlot et al. [3]. Section 4 explains the design of the Multi-neighborhood Multi-Operator algorithm, Section 5 provides benchmarking results followed by the conclusions in Section 6.

2. PROBLEM DESCRIPTION

The goal of exam timetabling is to obtain a schedule where each exam is allocated to an available timeslot such that the schedule satisfies all required constraints. An exam timetabling problem can be modeled by a labeled graph $\Gamma = (\mathbf{V}, \mathbf{E}, \gamma, \phi)$ where the vertices $\mathbf{V} = \{v_1, v_2, \ldots, v_{\|\mathbf{V}\|}\}$ represents the set of exams and the set of edges $\mathbf{E}$ represents the enrolment pattern of the students. An edge (v_i, v_j) exists if there is at least one student enrolled in exams v_i and v_j. The function $\gamma : \mathbf{V} \times \mathbf{V} \rightarrow \aleph$ returns the label of an edge. The label of an edge $(v_i, v_j) \in \mathbf{E}$ is a non-negative integer indicating the total number of students enrolled in the exams v_i and v_j. If $(v_i, v_j) \notin \mathbf{E}$ then $(v_i, v_j) = 0$. It is assumed that the edge labeling is a known problem parameter. The function $\phi : \mathbf{V} \rightarrow \aleph^+$ computes the label of a vertex. A vertex's label $\phi(v)$ is the timeslot number assigned to the exam v and is not known a priori. The timeslot numbering is assumed to be realized by a sequence of positive integers starting from unity. An exam timetabling is thus the labeling of the vertices such that a given criterion is optimized and the required constraints are satisfied. The resulting labeled graph is called a timetable graph. A timetable H is called feasible if it satisfies all constraints. Otherwise H is identified as unfeasible. Only feasible timetables are considered in this work.

A fundamental requirement in exam timetabling is to prohibit clashing or exam conflicts (a student having to take 2 or more exams in a given timeslot). Clashing is usually a hard constraint and can be expressed as

$$c_1: \phi(v_i) \neq \phi(v_j), \quad \forall (v_i, v_j) \in \mathbf{E}. \tag{1}$$

For practical reasons, a timetable must have a finite length. Since the timeslots are numbered contiguously, the constraint on the timetable length is,

$$c_2: \max_{v \in \mathbf{V}} \phi(v) \leq \rho, \tag{2}$$

where ρ is the largest allowable timeslot number, and is usually a problem parameter. Equations (1) and (2) are to be considered as hard constraints.

An exam proximity problem arises when the objective is to label the vertices of Γ such that the exams are as spread out as possible for all students. If the problem constraints exclude the maximum sitting capacity requirement then the resulting optimization problem is called the Uncapacitated Exam Proximity Problem (UEPP). For the UEPP, an often used objective function is the one proposed by Carter et al. [1]. In their proposal, penalties are given to timetable according to the number of timeslots between successive exams of each student. The overall penalty of the timetable is then averaged by the number of students taking part in the exams. More concisely, the UEPP minimization problem can be expressed as

$$\min f = \frac{1}{R} \sum_{\forall (v_i, v_j) \in \mathbf{E}} u(v_i, v_j), \tag{3}$$

subject to constraints c_1 (Eq. (1)) and c_2 (Eq. (2)). In Eq. (3), R is the total student enrolment, and $u : V \times V \rightarrow \aleph$ is a piece-wise penalty function, called proximity costs in [1], defined by

$$u(s,t) = \begin{cases} 2^{5-|\phi(s)-\phi(t)|}, & \text{if } 1 \leq |\phi(s)-\phi(t)| \leq 5; \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

Recall that $\phi(v)$ is the timeslot number assigned to the exam v. In summary, the spread of the exams is profitable from a student-centric point of view. It gives the students more time to prepare for their exams in a less stressful atmosphere. The next section details the enrolment datasets used in this work and surveys some previously published solution methods.

3. DATASETS AND PREVIOUS METHODS

There exists a collection of datasets that are available for benchmarking purposes. The datasets identified in Table 1 are the ones used by many researchers. They are actual enrolment data taken from several faculties and academic institutions.

Table 1: Dataset characteristics.

Dataset	Nb. of exams, $\|\|V\|\|$	Nb. of students, R	Exam enrolment
YOR-F-83	181	941	6034
TRE-S-92	261	4360	14901
KFU-S-93	461	5349	25113
MEL-F-01	521	20656	62247
MEL-S-01	562	19816	60637
NOT-F-94	800	7896	33997
PUR-S-93	2419	30032	120681

Datasets NOT-F-94 is by Burke et al. [2], MEL-F-01 and MEL-S-01 are from Merlot et al. [3]. The other datasets are from Carter et al. [1]. The number of exams, the number of students and the problem graph vary from one dataset to another and are not necessarily correlated. The challenge is to devise a timetabling algorithm that is uniformly effective for all datasets.

Early solution techniques were derived from sequential graph coloring heuristics. They were designed to assign each exam to a timeslot according to some ordering schemes. Carter et al. [1] successfully applied a backtracking sequential assignment algorithm to produce feasible UEPP timetables. Note that backtracking sequential assignment is a deterministic algorithm. It means that for a given dataset and ordering scheme, it will always produce the same timetable. In Burke et al. [2] an initial pool of timetable was generated by grouping together exams with similar sets of conflicting exams via a decomposition procedure. Then timetables were randomly selected from the pool, weighted by their objective value, and mutated by rescheduling randomly chosen exams. Hill climbing was then applied to the mutated timetable to improve its quality. Caramia et al. [4] developed a set of heuristics to tackle the UEPP with interesting results. First a solution was obtained by a greedy assignment procedure. This assignment procedure selects exams based on a priority scheme that gives high priority to exams with high clashing potential. Next, a spreading heuristic was applied to decrease the solution's proximity penalty without extending the timetable length. However, if the spreading heuristic failed to provide any penalty decrease then another heuristic is applied to decrease the proximity penalty by adding one extra timeslot to the solution. A perturbation technique was also described in which the search process was restarted by resetting the assigned priority.

A three-stage approach using constraint programming, simulated annealing and hill climbing was proposed by Merlot et al. [3] where an initial timetable was generated by constraint programming. The resulting timetable was then improved by a simulated annealing algorithm using the Kempe chain interchange neighborhood [5, 6] and a slow cooling scheme. In the last stage, a hill climbing search was applied to further improve the final timetable. Burke and Newall [7] investigated the effectiveness of local search approach to improve timetable quality. In their work, an adaptive technique was used to modify a given heuristic ordering for the sequential construction of an initial solution. They then compared the average and peak improvement obtained by three different search algorithms: Hill Climbing, Simulated Annealing and an implementation of the Great Deluge algorithm [8]. The reported results indicated that the combined adaptive heuristics and Great Deluge provided significant enhancement to the initial solution.

4. MULTI-NEIGHBORHOOD MULTI-OPERATOR ALGORITHM

In their review of metaheuristics for combinatorial optimization, Blum and Roli stated that every heuristic approach should be designed with the aim of effectively and efficiently exploring a search space [9]. A metaheuristic should both explore areas of the search space with high quality solutions, and to search unexplored areas when necessary. Blum and Roli also defined a general search method comparison framework based on the intensification and diversification concepts. According to this framework, intensification and diversification are search strategies that rely on randomness and the usage of memory. In an intensification strategy, the search should focus on exploring neighbors of good solutions. In a diversification strategy, the search should generate new solutions by visiting unexplored areas of the search space. Thus, to achieve an effective and efficient exploration of the search space, metaheuristic algorithms should be designed so that intensification and diversification play balanced roles [10].

The multi-neighborhood and multi-operator algorithm (MNMO) is an attempt to realize effective interplay between intensification and diversification during the search process. It is inspired by Hansen and Mladenović's Variable Neighborhood Descent (VND) algorithm [11] and the "one operator, one landscape" point of view advocated by Jones [12]. In the VND, a set of neighborhood structures is used sequentially. A solution is randomly selected in the k^{th} neighborhood structure using the current solution and becomes the starting point of a local search procedure. If an improvement is found, the algorithm starts again with $k = 1$ and the improved solution as the current solution. If there is no improvement, the algorithm proceeds to the $k + 1^{th}$ neighborhood structure using the previously recorded best solution as the current solution. The VND algorithm terminates when all neighborhood structures are visited and no improvement can be found by the local search procedure.

On the other hand, the "one operator, one landscape" view emerges from the observation that a search landscape is largely determined by the neighborhood structure. Thus, different neighborhood structures induce different search landscapes. In other words, a locally optimal solution in one neighborhood structure may not be so in another neighborhood structure. This idea is embedded in the VND metaheuristic as a diversification

strategy. However, the VND intensification strategy relies solely on a single local search operator and the intensification effort is somewhat imbalanced relative to the diversification effort. In order to correct this imbalance, multiple local search operators should be used. In the MNMO, each neighborhood structure is associated with one local search operator. The degree of intensification is thus increased by searching each neighborhood structure instead of randomly selecting a solution. Similar to most metaheuristic algorithms, the MNMO is iterative in nature. To avoid possible ambiguity the term "passes" will be used to indicate MNMO iterations. Figure 1 details the MNMO in pseudo-code form.

```
Algorithm MNMO
N_k(·) : k^th neighborhood structure
LS_k(·) : k^th local search operator
f(·) : objective function
Inputs
    Γ: problem graph; H : current timetable
    z_max : maximum non improving MNMO pass
    k_max : last local search and neighborhood structure index
Output
    B : current best timetable
H ← swo(Γ)
B ← H
z ← 0
while z < z_max {
    k ← 0
    while k < k_max {
        H' ← LS_k(N_k(H))
        if f(H') < f(H)
            H ← H'
        k ← k + 1
    }
    if f(H) < f(B) {
        B ← H
        z ← 0
    } else
        z ← z + 1
}
```

Figure 1: Multi-neighborhood Multi-Operator algorithm.

As shown in Figure 1, the starting point of a local search operator is the best timetable obtained by the previous local search operator. This is a simplified implementation of Glover and Laguna's Elite Restart Intensification strategy [13]. Here, a MNMO pass is the sequential execution of the k_{max} local search operators. The stopping criterion of the algorithm is based on the number of non-improving MNMO passes z_{max}. Finally, the search algorithm also memorizes the best timetable found in order to accommodate non greedy local search operators.

4.1 Initial timetables

The starting point of the MNMO algorithm is the generation of an initial feasible timetable. This is accomplished by an application of the so-called Squeaky Wheel Optimization (SWO) metaheuristic of Joslin and Clements [14]. In essence, the SWO is a parameterless three-phase iterative procedure. It begins with the construction of a timetable by labeling the exams in random order. Then it computes a priority, for each exam, based on the amount of constraint violations. Finally, the exams are reordered according to their priority. The construction – prioritization – reordering cycle continues until a stopping criterion is satisfied. The idea behind SWO is to discover an ordering of the exams by analyzing the appropriateness of the previous construction.

```
Algorithm SWO
L : exam priority list
Input
    Γ : problem graph
Output
    H : feasible timetable
H ← Γ
L ← random({v_1, …, v_||V||})
i ← true
while (i == true) {
    H ← construction(H, L)
    L ← prioritization(H)
    L ← sort(L, >)
    if (violations(H) == 0)
        i ← false
}
```

Figure 2: Squeaky Wheel Optimization algorithm.

In this algorithm the construction is realized by a simple sequential labeling of the exams in H. The priority list L determines the order in which the exams are selected for labeling. Initially, the ordering is random and an exam is labeled by selecting a timeslot number $1 \le p \le \rho$ with the smallest constraint violations. The sequential labeling may produce an unfeasible timetable and a new ordering will be required. To obtain a reordering, a prioritization routine is used to compute the amount of constraint violations of each exam. The priority $\tau(v_i)$ of exam v_i is the number of exam conflicts (constraint c_1, section 2) produced by its labeling. That is

$$\tau(v_i) = \begin{cases} \dfrac{\|\{v|\phi(v) = \phi(v_i) \wedge (v_i, v) \in \mathbf{E}\}\|}{\|\{v|(v_i, v) \in \mathbf{E}\}\|}, & \text{if } \|\{v|(v_i, v) \in \mathbf{E}\}\| > 0; \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

Exams with constraint violations should be label earlier than the ones without constraint violation. To achieve this, the exams are sorted according to their priority in decreasing order and the algorithm cycles through its phases until a feasible timetable is found.

4.2 Neighborhood Structures

Given a search space Ω consisting of all feasible timetables, a neighborhood structure is a function $\mathcal{N}: \Omega \to 2^{\Omega}$ that assigns to every $H \in \Omega$ a set of neighbors $\mathcal{N}(H) \subseteq \Omega$. Three neighborhood structures are used in the MNMO algorithm. Their definitions are given below.

Kempe Chain Interchange Neighborhood

The Kempe chain interchange neighborhood is defined by an ordered triple (v, p_0, p_1) where exam v is assigned to timeslot p_0 in the current timetable H and $p_0 \neq p_1$ [5, 6]. A Kempe chain (v, p_0, p_1) is a connected subgraph of H induced by the exams having timeslots p_0 and p_1. In other words, it is the set of exams reachable from v in the digraph Δ given by

$$V(\Delta) = \{V_{p_0}\} \cup \{V_{p_1}\}, \quad (6)$$

$$E(\Delta) = \left\{(u,w)|(u,w) \in E(\mathrm{H}), u \in V_{p_i} \wedge w \in V_{p_{(i+1) \bmod 2}}\right\},$$

where $E(\mathrm{H})$ represents the set of edges in the timetable graph and V_{p_i} is the subset of exams labelled with the timeslot number p_i that are reachable from exam v in the current timetable. Thus, a Kempe chain interchange is the relabeling of exams in Δ from timeslot p_0 to the timeslot p_1 and vice versa. Using the definition of Δ in Eq. (6) and denoting the relabelling operation by $V(\Delta)_{p_0} \rightleftarrows V(\Delta)_{p_1}$, the Kempe chain interchange neighborhood structure is the set of timeslots defined by

$$N_1(\mathrm{H}) = \begin{cases} V\left(\{V(H)\backslash V(\Delta)\} \cup \{V(\Delta)_{p_0} \rightleftarrows V(\Delta)_{p_1}\}\right), & \forall v_i, v_i \in \mathrm{H}, \\ & \phi(v_i) \neq \phi(v_j), \\ E(\mathrm{H}). & \end{cases} \quad (7)$$

2-exchange Neighborhood

The 2-exchange neighborhood is a commonly used structure. It is induced by exchanging the timeslot numbers of two exams in the current timetable.

$$N_2(\mathrm{H}) = \begin{cases} V\left(\left\{V(H)\backslash\{v_i \cup v_j\}\right\} \cup \{v_i \rightleftarrows v_j\}\right), & \forall v_i, v_i \in \mathrm{H}, \\ & \phi(v_i) \neq \phi(v_j), \\ E(\mathrm{H}). & \end{cases} \quad (8)$$

In Eq. (7), $v_i \rightleftarrows v_j$ signifies the exchange of $\phi(v_i)$ with $\phi(v_j)$.

1-interchange Neighborhood

This neighborhood structure is the simplest of the three. It is the assignment of a timeslot number p to an exam in the current timetable. The 1-interchange neighborhood is defined as

$$N_3(\mathrm{H}) = \begin{cases} V\left(\{V(H)\backslash\{v\}\} \cup \{v \leftarrow p\}\right), & \forall v \in \mathrm{H}, 1 \leq p \leq \rho, \\ & \phi(v) \neq p, \\ E(\mathrm{H}), & \end{cases} \quad (9)$$

where $v \leftarrow p$ denotes the timeslot assignment of exam v.

4.3 Local Search Operators

Each of the neighborhood structures described in section 4.2 has a corresponding local search operator. They are: *i*) Threshold Accepting algorithm (TA); *ii*) Record-to-Record Travel algorithm (RRT); *iii*) Tabu Search algorithm (TS). The TA and RRT algorithms proposed by Ducek and Scheuer are annealing-based general purpose optimization heuristics [8, 15]. In fact, TA and RRT are simplified variants of the Simulation Annealing (SA) metaheuristic. Pepper et al. compared several annealing-based optimization heuristics including the TA and RRT algorithms using the TSP problems [16]. Their findings indicate acceptable performance for both the TA and RRT. The well-known TS algorithm by Glover [13] is an often used metaheuristic for combinatorial problems. Its application and effectiveness in solving exam timetabling problems are already demonstrated by numerous technical publications.

5. EXPERIMENTAL RESULTS

The MNMO algorithm was evaluated using the datasets described in section 3. Table 2 presents the algorithmic parameters and computing environment used in the experiments. The algorithmic parameters were determined by applying the uncapacitated cobjective function f (Eq. 3) to the MEL-S-01 dataset (562 exams, 19816 students and 60637 exam enrolments). The parameter set tuning was performed by seeking a compromise between the quality of the solutions and the overall computation time. To simplify the choice of the neighborhood sampling size η of the local search algorithms, η was made to increase after a number of consecutive non improving iterations. Starting from an initial sampling size η_{min} it will increase by an amount equal to η_{min} up to a maximum sampling size of η_{max}. For the neighborhood structures, the local search – neighborhood structure pairings of Table 2 appeared to produce the best timetable proximity costs. The same MNMO parameters were used on all datasets.

Table 2: MNMO parameters and environment settings.

Local search – Neighborhood pairings	
LS_1: Threshold Accepting (TA)	Kempe chain interchange, $N_1(\mathrm{H})$
LS_2: Record-to-Record Travel (RRT)	2-exchange, $N_2(\mathrm{H})$
LS_3: Tabu Search (TS)	1-interchange, $N_3(\mathrm{H})$
Local search parameters	
Maximum iterations, i_{max}	40 000
Initial neighborhood sampling size, η_{min}	10
Final neighborhood sampling size, η_{max}	200
TA schedule	$T_0 = 0.5$, $T_f = 10^{-5}$
Tabu tenure	$t_{min} = 10$, $t_{max} = 35$
RRT deviation fraction	$d_f = 0.0075$
MNMO parameters	
Number of runs	5 per dataset
Number of non-improving passes	1
Computing environment	
Processor, RAM	Intel Core i7, 3.4 GHz, 8 GB
OS	Windows 7 Enterprise
Compiler and optimization level	VC++, Release Mode

5.1 UEPP Results

The results for the UEPP are summarized in Table 3. Feasible timetables were obtained in all cases for all runs. On average, the computation time varied from 2 minutes to 9 hours per run.

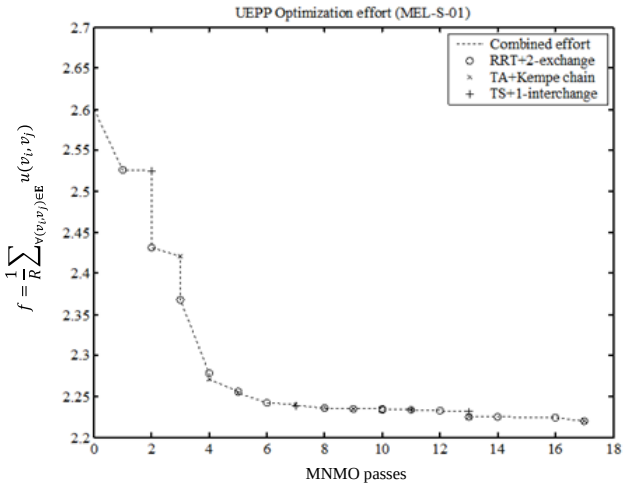

Figure 3: Minimization of f (MEL-S-01 dataset)

Fig. 3 depicts the optimization effort produced by the MNMO algorithm on the MEL-S-01 dataset during one MNMO run. A "relaying" effect among the local search operators is observed during the optimization process. An improved timetable generated by one local search operator is further improved by another search operator in a way that is analogous to a track and field relay team. However, the optimization process continues even if a local search operator failed to improve a given timetable. The timetable is simply passed on to the next search operator. This effect can be discerned in pass 2, 3 and 4 of Fig. 3.

Table 3: Results for the UEPP.

Dataset	Timeslots, ρ		f_1	Run time
YOR-F-83	21	best	36,2	
		avg.	36,7	2 min.
TRE-S-92	23	best	8,1	
		avg.	8,3	4 min.
KFU-S-93	20	best	13,0	
		avg.	13,3	60 min
MEL-F-01	28	best	2,7	
		avg.	2,8	27 min
MEL-S-01	31	best	2,2	
		avg.	2,3	30 min
NOT-F-94	23	best	6,4	
		avg.	6,5	120 min
PUR-S-93	42	best	4.7	
		avg.	5.0	9 h

Table 4 compares the performance of the MNMO algorithm against other solution methods. It is observed that, for small size datasets, Caramia, Burke, Merlot and MNMO produced the best results. For the largest dataset (PUR-S-93) in the collection, only Carter and Caramia were able to produce acceptable results. The MNMO algorithm performed quite well for all datasets, obtaining the best proximity costs for 6 datasets of different sizes.

Dataset		MNMO	Carter [1]	Caram [4]	Burke [7]	Merlot [3]
YOR-F-83	Best	**36.2**	36.4	**36.2**	36.8	37.4
	Avg	36.7	45.6	-	37.3	37.9
TRE-S-92	Best	**8.1**	9.6	9.4	8.2	8.4
	Avg	8.3	10.8	-	8.4	8.6
KFU-S-93	Best	**13.0**	14.0	13.8	13.7	13.5
	Avg	13.3	18.8	-	13.9	14.0
MEL-F-01	Best	**2.7**	-	-	-	2.9
	Avg	2.8	-	-	-	3.0
MEL-S-01	Best	**2.2**	-	-	-	2.5
	Avg	2.3	-	-	-	2.5
NOT-F-94	Best	**6.4**	-	-	-	7.0
	Avg	6.5	-	-	-	7.1
PUR-S-93	Best	4.7	3.9	**3.7**	-	-
	Avg	5.0	4.4	-	-	-

6. CONCLUSIONS

The MNMO algorithm performed well in comparison to several other solution methods. For the UEPP it was able to match the best proximity costs for 7 datasets. These results should encourage further research on the application of multi-search paradigm as solution methods to the exam timetabling problem. However, the cost of such methods is rather expensive. As shown in this research, the main drawback of the multi-neighborhood multi-search technique is the high computation load involved in the optimization process. It is possible to lower the temporal complexity by pruning the search space as suggested by Prestwich [17]. Another complexity reducing technique is to distribute the workload to multiple processing units. By decomposing the timetabling problem into several smaller subproblems as in [2], it is possible to solve the problem instances in parallel. However, more research is needed to access the effectiveness of these auxiliary techniques.

7. REFERENCES

[1] M. Carter, G. Laporte, and S.T. Lee, Examination timetabling: algorithmic strategies and applications, **Journal of the Operational Research Society**, Vol. 47, 1996, pp. 373-383.

[2] E.K. Burke, J. Newall, and R.F. Weare, A memetic algorithm for university exam timetabling. In: Burke, E.; Ross, P. (eds.): Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, U.K., August/September 1995. Selected Papers. **Lecture Notes in Computer Science 1153**, Springer-Verlag, Heidelberg, 1996, pp. 241-250.

[3] L.T.G. Merlot, N. Boland, B.D. Hughes and P.J. Stuckey, A Hybrid Algorithm for the Examination Timetabling Problem. In: Burke, E.; De Causmaecker, P. (eds.): Practice and Theory of Automated Timetabling, Fourth International Conference, Gent, Belgium, August 2002. Selected Papers. **Lecture Notes in Computer Science 2740**, Springer-Verlag, Heidelberg, 2003, pp. 207-231.

[4] M. Caramia, P. Dell'Olmo, and G.F. Italiano, New algorithms for examination timetabling. In: Nher, S.; Wagner, D., (eds.): Algorithm Engineering 4th International Workshop, WAE 2000, Saarbrucken, Germany, September 2000. Proceedings. **Lecture Notes in Computer Science 1982**, Springer-Verlag, Heidelberg, 2001, pp. 230-241.

[5] C. Morgenstern, H. Shapiro, Coloration neighborhood structures for general graph coloring. In **Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms**, San Francisco, California, 1990, pp. 226-235.

[6] J. Thompson, K. Dowsland, A robust simulated annealing based examination timetabling system. **Computers and Operations Research**, vol. 25, 1998, pp. 637-648.

[7] E.K. Burke, J. Newall, Enhancing Timetable Solutions with Local Search Methods. In: Burke, E.; De Causmaecker, P. (eds.): Practice and Theory of Automated Timetabling, Fourth International Conference, Gent, Belgium, August 2002. Selected Papers. **Lecture Notes in Computer Science 2740**, Springer-Verlag, Heidelberg, 1996, pp. 195-206.

[8] G. Dueck, New Optimization Heuristics. The Great Deluge Algorithm and the Record-to-Record Travel. **Journal of Computational Physics**, vol. 104, 1993, pp. 86-92.

[9] C. Blum, A. Roli., Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. **ACM Computing Surveys**, vol. 35, 2003, pp. 268-308.

[10] M. Yagiura, T. Ibaraki, On metaheuristic algorithms for combinatorial optimization problems. **Syst. Comput. Japan 32**, vol. 3, 2001, pp. 33-55.

[11] N. Mladenovic, P. Hansen. Variable Neighborhood

Search, **Computers and Operations Research**, vol. 24, 1997, pp. 1097-1100.

[12] T. Jones, *One operator, one landscape*. Santa Fe Institute Technical Report 95-02-025, Santa Fe Institute, 1995. (Downloadable from http://www.santafe.edu/media/workingpapers/95-02-025.pdf)

[13] F. Glover, F. Laguna, **Tabu Search**, Kluwer Academic Publishers, Norwell, 1997.

[14] D. E. Joslin, D. P. Clements, Squeaky Wheel Optimization, **Journal of Artificial Intelligence Research**, vol. 10, 1999, pp. 353-373.

[15] G. Dueck, T. Scheuer, Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing, **Journal of Computational Physics**, vol. 90, 1990, 161—175.

[16] J.W. Pepper, B. L. Golden, E. A. Wasil, Solving the traveling salesman problem with annealing-based heuristics: a computational study, **IEEE Systems, Man and Cybernetics**, Part A, vol. 32, 2002, 72—77.

[17] S. Prestwich. Combining the scalability of local search with the pruning techniques of systematic search. **Annals of Operations Research**, vol. 115, 2002, pp. 51-72.