2005 IEEE International Conference on Systems, Man and Cybernetics
Waikoloa, Hawaii October 10-12, 2005

A Multi-Neighborhood and Multi-Operator Strategy for the Uncapacitated Exam Proximity Problem

Tony Wong, Pascal Bigras
Automated Manufacturing Engineering Department
École de technologie supérieure
Montréal, Québec, Canada
{tony.wong, pascal.bigras}@etsmtl.ca

Bruno de Kelper
Electrical Engineering Department
École de technologie supérieure
Montréal, Québec, Canada
bruno.dekelper@etsmtl.ca

Abstract – *A strategy featuring multiple local search operators and multiple neighborhood structures is applied to the 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 the 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 it was able to produce the best proximity costs for several datasets.*

Keywords: Timetabling, metaheuristic, multiple neighborhoods, local search, benchmarking.

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 classroom sitting capacity. Usually, there exists a set of requirements that determine the timetable characteristics. For example, a timetable must avoid scheduling students to more than one exam per timeslot. Also, it is desirable for a timetable to have a prescribed length. Thus, the timetable construction is a process that optimizes the proximity of the exams while satisfying the given constraints. In this work, a strategy 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 optimization effort is enhanced. The result is a multi-operator optimization 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 the graph-theoretic approach. Section 3 presents the datasets used in the benchmarking of the UEPP 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 details the benchmarking results followed by the conclusions in section 6.

2 Problem description

The goal of exam timetabling is to obtain an assignment where each exam is allocated to an available timeslot such that the assignment satisfies all required constraints. An exam timetabling problem can be modeled by a labeled graph $\mathcal{G} = (\mathbf{V}, \mathbf{E}, \gamma, \phi)$ 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 $\gamma(v_i, v_j) = 0$. 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 the vertex labeling 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 problem is thus the labeling of the vertices such that a given criterion is optimized and the required constraints are satisfied. Conceptually the result of the vertex labeling is a graph called a timetable graph or simply a timetable.

A timetable is structurally identical to the problem graph $\mathcal{G}$ but has a different vertex labeling. For a given timetable H, $V(\mathrm{H}) = \mathbf{V}$ denotes the set of exams within H. 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,

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

where ρ is the largest allowable timeslot number, and is usually a known 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 $\mathcal{G}$ such that the exams are as spread out as possible for all students. It is called the Uncapacitated Exam Proximity Problem (UEPP) if the classroom sitting capacity is excluded from the constraint set. For the UEPP, an often used objective function is the one proposed in [1]. In that proposal, penalties are given to the 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

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

subjected to the constraints c_1 (Eq. (1)) and c_2 (Eq. (2)). In equation (3), R is the total student enrolment and $u : \mathbf{V} \times \mathbf{V} \rightarrow \aleph$ is a piece-wise penalty function defined by

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

In summary, the spreading 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 research 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 I are the ones used by many researchers. They are actual enrolment data taken from several universities and academic institutions. Dataset NOT-F-94 is from [2], MEL-F-01 and MEL-S-01 are from [3]. The other datasets are given in [1]. The number of exams $||\mathbf{V}||$, the number of students R 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 are derived from sequential graph coloring heuristics. They are 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 timetables for the UEPP. The backtracking feature enables the algorithm to undo previous assignments and thus escape from cul-de-sacs. In all, 40 different strategies were implemented. The results showed that the effectiveness of the sequential assignment algorithm is related to the ordering scheme and the nature of the datasets.

Table I. Datasets characteristics

Dataset	$\|\|\mathbf{V}\|\|$	R	Enrolment
HEC-S-92	81	2823	10632
STA-F-83	139	611	5751
YOR-F-83	181	941	6034
UTE-S-92	184	2749	11793
EAR-F-83	190	1125	8109
TRE-S-92	261	4360	14901
LSE-F-91	381	2726	10918
KFU-S-93	461	5349	25113
RYE-S-93	486	11483	45051
MEL-F-01	521	20656	62247
CAR-F-92	543	18419	55522
MEL-S-01	562	19816	60637
UTA-S-92	622	21266	58979
CAR-S-91	682	16925	56877
NOT-F-94	800	7896	33997
PUR-S-93	2419	30032	120681

In Burke et al. [2] an initial pool of timetables is generated by grouping together exams with similar sets of conflicting exams via a decomposition procedure. Then timetables are randomly selected from the pool, weighted by their objective value, and mutated by rescheduling randomly chosen exams. Hill climbing is then applied to the mutated timetable to improve its quality. The process continues with the new pool of timetables. Caramia et al. [4] developed a set of heuristics to tackle the UEPP with interesting results. First a solution is 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 is applied to decrease the proximity penalty of the solution without extending the timetable length. 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. These heuristics are reapplied until no further improvement can be found. The proximity problem was also investigated by Di Gaspero and Scharef [5]. Their approach starts with a greedy heuristic to assign timeslots to all exams that have no common students. The remaining unassigned exams are distributed randomly to different timeslots. The obtained solution is then improved by a Tabu Search algorithm using a short-term Tabu list with random Tabu tenure. The search neighborhood is defined as the set of exams that can be moved from one timeslot to another without violating the constraints. A further reduction of the neighborhood is obtained by using the subset of exams that are currently in constraint violation. A Tabu Search algorithm equipped with a recency-based and a frequency-based Tabu list was implemented by White and Xie [6]. An initial solution is first generated by a bin packing heuristic ("largest

enrolment first"). If the initial solution is unfeasible then a Tabu Search is executed to remove all constraints violations using as neighborhood the set of clashing exams. Another Tabu Search is used to improve the quality of the feasible solution. More recently Paquete and Stutzle [7] considered the UEPP by casting the constraints as part of an aggregated objective function. The search process is prioritized and is realized by the use of a Tabu Search algorithm with short-term Tabu list and random tenure. The 1-opt neighborhood is defined by the subset of exams with constraint violations.

A three-stage approach using constraint programming, simulated annealing and hill climbing was proposed by Merlot et al. [3]. An initial timetable is generated by constraint programming. The resulting timetable is then improved by a simulated annealing algorithm using the Kempe chain interchange neighborhood [8, 9] and a slow cooling schedule. In the last stage, a hill climbing procedure is applied to further improve the final timetable. The GRASP metaheuristic [10] was also used to solve the UEPP. Casey and Thompson [11] used a probabilistic version of the sequential assignment algorithm from [1] to realize the construction phase of GRASP. In the improvement phase of GRASP, they ordered the exams according to their contribution to the objective value. Then for each exam a timeslot is found such that the objective value is decreased. The construction and improvement phases are restarted with a blank timetable for a number of times and the best timetable is kept. Their best results were obtained by a saturation degree construction and an improvement using the Kempe chain interchange neighborhood.

Burke and Newall [12] investigated the effectiveness of local search approach to improve the quality of timetables. In their work, an adaptive technique is 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 [13]. The reported results indicated that the combined Adaptive Heuristics and Great Deluge provided significant enhancement to the initial solution.

4 Neighborhoods & local search

In their review of metaheuristics for combinatorial optimization, Blum and Roli stated that every metaheuristic approach should be designed with the aim of effectively and efficiently exploring a search space [14]. A metaheuristic should both explore areas of the search space with high quality solutions, and to search unexplored areas of the search space 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 [15].

The Multi-Neighborhood Multi-Operator strategy (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 [16] and the "one operator, one landscape" point of view advocated by Jones [17]. 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+1th 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 strategy in algorithmic form. 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 [18]. 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.

Algorithm MNMO
$\mathcal{N}_k(\cdot)$: k^{th} neighborhood structure; $LS_k(\cdot)$: k^{th} local search operator
$f(\cdot)$: objective function
Inputs
$\mathcal{G}$: problem graph; H : current timetable
z_{max} : maximum non improving MNMO passes
k_{max} : last local search and neighborhood structure index
Output
B : current best timetable

```
H ← swo(G)
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

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 [19]. 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. In SWO the construction is realized by a simple sequential labeling of the exams in H. A 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. For our problem, the priority $\wp(v_i)$ of exam v_i is the number of exam conflicts (constraint c_1, section 2) produced by its labeling. That is $\wp(v_i) = \tau_1(v_i)$ where

$$\tau_1(v_i) = \begin{cases} (\| \{v \mid \phi(v) = \phi(v_i) \wedge (v_i, v) \in \mathbf{E}\} \|) / \left\| \{v \mid (v_i, v) \in \mathbf{E}\} \right\| & \text{if } \left\| \{(v \mid (v_i, v) \in \mathbf{E}\} \right\| > 0; \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

Exams with constraint violations should be labeled 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 $\mathcal{H}$ consisting of all feasible timetables, a neighborhood structure is a function $\mathcal{N} : \mathcal{H} \to 2^{\mathcal{H}}$ that assigns to every $H \in \mathcal{H}$ a set of neighbors $\mathcal{N}(H) \subseteq \mathcal{H}$. Three neighborhood structures are used in the MNMO algorithm. Their definitions are given in this subsection.

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 $p_0 \neq p_1$ [9, 10]. 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 $\mathcal{D}$ given by

$$\begin{aligned} V(\mathcal{D}) &= \left\{V_{p_0}\right\} \cup \left\{V_{p_1}\right\}, \\ E(\mathcal{D}) &= \left\{(u,w) \mid (u,w) \in E(\mathrm{H}), u \in V_{p_i} \wedge w \in V_{p_{(i+1) \bmod 2}}\right\}, \end{aligned} \quad (6)$$

where E(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 relabelling of exams in $\mathcal{D}$ from timeslot p_0 to the timeslot p_1 and vice versa. Using the definition of $\mathcal{D}$ in (6) and denoting the relabelling operation by $V(\mathcal{D})_{p_0} \rightleftarrows V(\mathcal{D})_{p_1}$, the Kempe chain interchange neighborhood structure is the set of timetables defined by

$$\mathcal{N}_1(\mathrm{H}) = \begin{cases} V(\{V(\mathrm{H}) \setminus V(\mathcal{D})\} \\ \cup \{V(\mathcal{D})_{p_0} \rightleftarrows V(\mathcal{D})_{p_1}\}), & p_0, p_1 = 1, 2, \ldots, \rho; \\ & p_0 \neq p_1; \\ E(\mathrm{H}). \end{cases} \quad (7)$$

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.

$$\mathcal{N}_2(\mathrm{H}) = \begin{cases} V(\{V(H) \setminus \\ \{v_i \cup v_j\}\} \cup \{v_i \rightleftarrows v_j\}), & \forall v_i, v_j \in \mathrm{H}; \\ & \phi(v_i) \neq \phi(v_j); \\ E(\mathrm{H}). \end{cases} \quad (8)$$

In equation (8), $v_i \rightleftarrows v_j$ 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

$$\mathcal{N}_3(\mathrm{H}) = \begin{cases} V(\{V(\mathrm{H})\backslash\{v\}\} \cup \{v \leftarrow p\}), & \forall v \in \mathrm{H}, 1 \le p \le \rho; \\ E(\mathrm{H}), & \phi(v) \ne p; \end{cases} \quad (9)$$

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 [20, 13]. In fact, TA and RRT are simplified variants of the Simulation Annealing (SA) metaheuristic. Recently, Pepper et al. compared several annealing-based optimization heuristics including the TA and RRT algorithms using the TSP problems [21]. Their results indicate acceptable performance for both the TA and RRT. The well-known TS algorithm by Glover [18] is an often used metaheuristic for combinatorial problems. Its application and effectiveness in solving exam timetabling problems are already demonstrated by numerous technical publications.

Threshold Accepting Algorithm The TA algorithm is a simplified variant of the SA algorithm. Similar to the SA, the TA algorithm accepts "uphill moves". However a move is accepted only if it produces an objective value less than a certain threshold. Here an uphill move signifies that the current timetable may be replaced by another timetable which has a higher proximity cost. Thus, the threshold, annealed according to a schedule, is the bound on the increase in proximity cost from one iteration to the next. The best improvement strategy is used to select a single timetable H' $\in$ **I**. If the difference in proximity cost between the current timetable H and the newly selected timetable H' is less than the threshold T then H' replaces H. Let i, $i_{\max}$ the current and the maximum iteration numbers, and T_0, T_f the initial and final threshold values. The threshold schedule can be defined as $T = T_0\left(T_f / T_0\right)^{i/i_{\max}}$, an exponentially decreasing function often used in Competitive Learning.

Record-to-Record Travel Algorithm The "Record" in this search algorithm is the proximity cost of the current best timetable. The RRT algorithm can only explore timetables with proximity cost that is lower than the sum of Record and a deviation value d_f. Together, the Record and the deviation serve as an upper bound on the proximity cost that can be accepted. Thus, the upper bound is a function of the best proximity cost found. The structure of the RRT algorithm is very similar to the TA algorithm. The main difference is in the computation of the upper bound. In the TA algorithm, the annealing schedule depends solely on the iteration counter. Here the upper bound is a function of the best timetable found so far.

Tabu Search Algorithm A simple TS algorithm is implemented in the MNMO. It is equipped with a Tabu list and a single aspiration criterion. The Tabu list acts as a short term memory in order to avoid cycles. The Tabu tenure is a bounded random value between $t_{\min}$ and $t_{\max}$. In practice, the Tabu list only records the difference in exam labeling between H and H'. This list is updated for every non Tabu timetables. The aspiration criterion allows a Tabu timetable to become the current timetable B if it has a better proximity cost than B.

5 Experimental Results

The MNMO algorithm was evaluated using the datasets described in section 3.

Table 2. MNMO parameters and environmental setting

Local search – Neighborhood pairings	
LS_1: Threshold Accepting (TA)	Kempe chain interchange, $\mathcal{N}_1$(H)
LS_2: Record-to-Record Travel (RRT)	2-exchange, $\mathcal{N}_2$(H)
LS_3: Tabu Search (TS)	1-interchange, $\mathcal{N}_3$(H)
Local search parameters	
Maximum iterations, $i_{\max}$	40 000
Initial neighborhood sampling size, $\eta_{\min}$	10
Final neighborhood sampling size, $\eta_{\max}$	200
Consecutive non improving iterations	$i_{\max}$
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	
System	Ahtlon XP 2.2 GHz, 2 GB RAM
OS	Windows XP, SP1

Table 2 presents the algorithmic parameters and computing environment settings used in the experiments. The algorithmic parameters were determined by applying the uncapacitated objective function f (Eq. (3)) to the MEL-S-01 dataset. 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 $\eta_{\min}$ it will increase by an amount equal to $\eta_{\min}$ up to a maximum sampling size of $\eta_{\max}$. 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. 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 4 minutes to more than 36 hours per run. Figure 2 depicts the combined optimization effort produced by the MNMO algorithm.

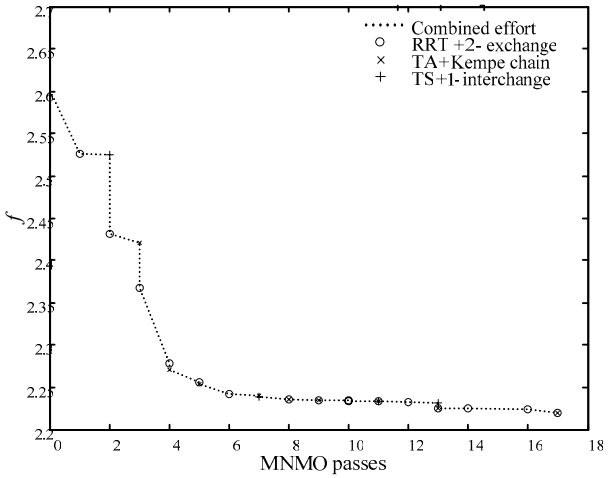

Figure 2. Minimization of proximity (MEL-S-01 dataset)

Table 3. Results for the UEPP

Dataset	Timeslots, ρ		*f*	Run time
HEC-S-92	18	best	10,1	
		avg.	10,3	4 min.
STA-F-83	13	best	157,0	
		avg.	157,0	4 min.
YOR-F-83	21	best	36,2	
		avg.	36,7	6 min.
UTE-S-92	10	best	24,9	
		avg.	25,0	5 min.
EAR-F-83	24	best	33,3	
		avg.	33,6	7 min.
TRE-S-92	23	best	8,1	
		avg.	8,3	15 min.
LSE-F-91	18	best	10,1	
		avg.	10,3	42 min.
KFU-S-93	20	best	13,0	
		avg.	13,3	2.2 h
RYE-S-93	23	best	7,4	
		avg.	7,5	1 h
MEL-F-01	28	best	2,7	
		avg.	2,8	1.8 h
CAR-F-92	32	best	4,0	
		avg.	4,1	2 h
MEL-S-01	31	best	2,2	
		avg.	2,3	2 h
UTA-S-92	35	best	3,7	
		avg.	3,7	4 h
CAR-S-91	35	best	5,2	
		avg.	5,3	4.2 h
NOT-F-94	23	best	6,4	
		avg.	6,5	6.2 h
PUR-S-93	42	best	4.7	
		avg.	5.0	36 h

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 easily discerned in pass 2, 3 and 4 of Figure 2. Table 4 compares the performance of the MNMO algorithm against other solution methods. It is observed that, for small size datasets, Caramia et al.'s heuristic approach produced the best results. For large size datasets, Burke and Newall's adaptive reordering technique achieved the best proximity costs. The MNMO algorithm performed quite well for all datasets, obtaining the best results for 7 datasets of different sizes. It is worth noting that only Carter et al. and Caramia et al. were able to produce acceptable results for the largest dataset (PUR-S-93, 2419 exams) in the collection. The MNMO algorithm was unable to compete against their methods even after 36 hours of computing time.

6 Conclusions

The MNMO algorithm performed well in comparison to several other solution methods. For the UEPP it was able to produce the best proximity costs for 7 datasets. However, the cost of such methods is rather expensive. As shown in this research, the main drawback of the multi-neighborhood multi-operator technique is the high computation load involved in the optimization process. It is possible to lower the temporal complexity by pruning the search space. 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.

References

[1] M. Carter, G. Laporte, and S.T. Lee, "Examination timetabling: algorithmic strategies and applications," *Journal of the Op. Res. Soc.*, vol. 47, pp. 373—383, 1996.

[2] E.K. Burke, J. Newall, and R.F. Weare, "A memetic algorithm for university exam timetabling," PATAT, *LNCS* 1153, Springer-Verlag, pp. 241—250, 1996.

[3] L.T.G. Merlot, N. Boland, B.D. Hughes and P.J. Stuckey, "A Hybrid Algorithm for the Examination Timetabling Problem," PATAT, *LNCS* 2740, Springer-Verlag, pp. 207—231, 2003.

[4] M. Caramia, P. Dell'Olmo, and G.F. Italiano, "New algorithms for examination timetabling," Int. Workshop on Algo. Eng., *LNCS* 1982, Springer-Verlag, pp. 230—241, 2001.

[5] L. Di Gaspero, A. Schaerf. "Tabu search techniques for examination timetabling," PATAT, *LNCS* 2079, Springer-Verlag, pp. 104—117, 2001.

[6] G.M. White, B.S. Xie, "Examination timetables and tabu search with longer-term memory," PATAT, *LNCS* 2079 Springer-Verlag, pp. 85—103, 2001.

[7] L. Paquete, T. Stutzle, "Empirical Analysis of Tabu Search for the Lexicographic Optimization of the Examination Timetabling Problem," *PATAT*, pp. 413—420, 2002.

Table 4. UEPP Benchmarks

Dataset		MNMO	Carter [1]	White [6]	DiGa [5]	Caram [4]	Burke [12]	Merlot [3]	Paquet [7]	Casey [11]
HEC-S-92	Best	10.1	10.6	-	12.4	**9.2**	11.3	10.6	11.2	10.8
18 timeslots	Avg	10.3	15.0	-	12.6	-	11.5	10.7	12.0	10.9
STA-F-83	Best	157.0	161.5	-	160.8	158.2	168.3	157.3	158.1	**134.9**
13 timeslots	Avg	157.0	167.1	-	166.8	-	168.7	157.4	159.3	135.1
YOR-F-83	Best	**36.2**	36.4	-	41.0	**36.2**	36.8	37.4	38.9	37.5
21 timeslots	Avg	36.7	45.6	-	42.1	-	37.3	37.9	41.7	38.1
UTE-S-92	Best	24.9	25.8	-	29.0	**24.4**	25.5	25.1	27.8	25.4
10 timeslots	Avg	25.0	30.8	-	31.3	-	25.8	25.2	29.4	25.5
EAR-F-83	Best	33.3	36.4	-	45.7	**29.3**	36.1	35.1	38.9	34.8
24 timeslots	Avg	33.6	40.9	-	46.7	-	37.1	35.4	42.0	35.0
TRE-S-92	Best	**8.1**	9.6	-	10.0	9.4	8.2	8.4	9.3	8.7
23 timeslots	Avg	8.3	10.8	-	10.5	-	8.4	8.6	10.2	8.8
LSE-F-91	Best	10.1	10.5	-	15.5	**9.6**	10.6	10.5	13.2	14.7
18 timeslots	Avg	10.3	12.4	-	15.9	-	10.8	11.0	15.5	15.0
KFU-S-93	Best	**13.0**	14.0	-	18.0	13.8	13.7	13.5	16.5	14.1
20 timeslots	Avg	13.3	18.8	-	19.5	-	13.9	14.0	18.3	14.3
RYE-S-92	Best	7.4	7.3	-	-	**6.8**	-	8.4	-	-
23 timeslots	Avg	7.5	8.7	-	-	-	-	8.7	-	-
MEL-F-01	Best	**2.7**	-	-	-	-	-	2.9	-	-
28 timeslots	Avg	2.8	-	-	-	-	-	3.0	-	-
CAR-F-92	Best	**4.0**	6.2	-	5.2	6.0	**4.0**	4.3	-	4.4
32 timeslots	Avg	4.1	7.0	4.7	5.6	-	4.1	4.4	-	4.7
MEL-S-01	Best	**2.2**	-	-	-	-	-	2.5	-	-
31 timeslots	Avg	2.3	-	-	-	-	-	2.5	-	-
UTA-S-92	Best	3.7	3.5	-	4.2	3.5	**3.2**	3.5	-	-
35 timeslots	Avg	3.7	4.8	4.0	4.5	-	3.2	3.6	-	-
CAR-S-91	Best	5.2	7.1	-	6.2	6.6	**4.6**	5.1	-	5.4
35 timeslots	Avg	5.3	8.4	-	6.5	-	4.7	5.2	-	5.6
NOT-F-94	Best	**6.4**	-	-	-	-	-	7.0	-	-
23 timeslots	Avg	6.5	-	-	-	-	-	7.1	-	-
PUR-S-93	Best	4.7	3.9	-	-	**3.7**	-	-	-	-
42 timeslots	Avg	5.0	4.4	-	-	-	-	-	-	-

[8] C. Morgenstern, H. Shapiro, Coloration neighborhood structures for general graph coloring. In *Proc. of the 1st ACM-SIAM Symp. on Dis. Algo.*, San Francisco, California, pp. 226—235, 1990.

[9] J. Thompson, K. Dowsland, "A robust simulated annealing based examination timetabling system," *Comp. and Op. Res.*, vol. 25, pp. 637—648, 1998.

[10] T.A. Feo, M.G.C. Resende, S.H. Smith, "A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set," *Op. Res.*, vol. 42, pp. 860—878, 1994.

[11] S. Casey and J. Thompson. "GRASPing the Examination Scheduling Problem," PATAT, *LNCS* 2740, Springer-Verlag, pp. 232—244, 1996.

[12] E.K. Burke, J. Newall, "Enhancing Timetable Solutions with Local Search Methods," PATAT, *LNCS* 2740, Springer-Verlag, pp. 195—206, 1996.

[13] G. Dueck, New Optimization Heuristics. "The Great Deluge Algorithm and the Record-to-Record Travel," *Journal of Comp. Phy.*, vol. 104, pp. 86—92, 1993.

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

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

[16] N. Mladenovic, P. Hansen. "Variable Neighborhood Search," *Comp. and Op. Res.*, vol. 24, pp. 1097—1100, 1997.

[17] T. Jones, *One operator, one landscape*. Santa Fe Institute Technical Report 95-02-025, Santa Fe Institute, 1995. (http://www.santafe.edu/sfi/publications/wpabstract/199502025).

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

[19] D. E. Joslin, D. P. Clements, "Squeaky Wheel Optimization," *Journal of AI Res.*, vol. 10, pp. 353—373, 1999.

[20] G. Dueck, T. Scheuer, "Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing," *Journal of Comp. Phy.*, vol. 90, pp. 161—175, 1990.

[21] 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, pp. 72—77, 2002.