SCHEDULING SCHOOL MEETINGS

Franca Rinaldi and Paolo Serafini

University of Udine, Department of Mathematics and Computer Science

Abstract: Prespecified meetings between teachers and parents have to be scheduled. All meetings have the same duration. The goal is in finding a schedule minimizing the total time and the parents' idle times. This NP-hard problem is addressed by solving first a sequence of weighted assignment problems and then performing a large scale neighborhood search based on finding negative cost cycles and shortest paths in directed graphs. This approach provides good computational results. Finally a variant of the problem with two different durations for the meetings is considered.

1. Introduction

We address the following problem arising in Italian high schools: on certain days of the school year, parents can meet teachers to discuss about their children. Each parent tries to meet some teachers and the meetings are individual. There is no advance planning for the event and therefore parents wait in lines for a long time (one line for each teacher), not only wasting time but also preventing the possibility to meet several teachers.

In this paper we propose a planning method in order to guarantee that each parent meets all required teachers and the global wasted time of the parents is minimized. A prerequisite for the method to work is that all meetings last the same amount of time.

The problem we study is NP-hard. We suggest a two phase heuristic approach for its solution. The first phase computes a schedule of minimum time to allocate all meetings by solving a sequence of weighted assignment problems. The second phase minimizes the parents' idle times through a large scale neighborhood search based on negative cycle detection and shortest path computations in directed graphs. It turns out that this search is quite effective and strongly reduces the wasted time of the final solutions.

We also briefly consider an extension of the model by allowing double duration to certain specified meetings that require more discussion.

To the best of our knowledge, this specific timetable problem has not been addressed in the literature. Nevertheless, a similar problem was considered in Bartholdi and McCroan (1990) to schedule job interviews for law firms and students at the Southeastern Public Interest Job Fair, a law fair which is held each year in the U.S. In this paper the timetable problem is modeled as an edge–coloring problem on bipartite graphs and the minimization of the idle times (considered both for law firms and students) is partially carried out by maximizing the cumulative number of meetings assigned to each period. Our problem has also some resemblance with the no–wait open shop problem (Hall and Sriskandarajah (1990)), where it is required that all the operations of a same job are executed contiguously, i.e., no idle time for the jobs is allowed. The particular case of unit processing times has been studied in Gonzalez (1982) and, under the additional condition of no idle time for the machines, in Giaro and Kubale (2004).

As pointed out above, the local search procedure we adopt is based on neighborhoods of exponential size in which an improving neighbor can be found in polynomial time by dynamic programming. The potential of using optimization methods, and in particular network flow techniques and dynamic programming, to search neighborhoods of very large size, has been remarked in Ahuja et al. (2002). In the particular field of timetabling, large–scale neighborhood metaheuristics have been applied, for instance, in Dowsland (1998) to solve a nurse scheduling problem and in Meyers and Orlin (2006) to solve the examination timetabling problem.

The paper is organized as follows. In Section 2 we define the problem. In Section 3 an algorithm to find a schedule of minimum makespan is presented. In Sections 4 and 5 two local search procedures based on finding sequences of meeting exchanges are described. The first one considers meetings of a fixed teacher, whereas the second one considers meetings of a fixed parent. The techniques are illustrated with an example in Section 6. In Section 7 we extend the problem to the case with two different durations of the meetings. Finally some computational results are given in Section 8.

2. Problem Definition

In the problem we consider, a set J of parents and a set I of teachers are given and each parent j wants to meet a specified subset I_j of teachers. The subsets I_j are the input data. Let $J_i := \{j : i \in I_j\}$ be the subset of parents that want to meet the teacher i.

Assuming that all meetings last the same amount of time, called *time slot*, the output is a schedule $t(i,j)$ assigning the time slot $t(i,j)$ to the meeting of parent j with teacher i, with the obvious requirement that $t(i,j) \neq t(i,j')$ for all $j' \neq j$ and all i and also $t(i,j) \neq t(i',j)$ for all $i' \neq i$ and all j. We define as *makespan* of the schedule the maximum time in which a meeting occurs, i.e., $\max_{i \in I} t(i,j) = \max_{j \in J} t(i,j)$. Moreover, for any given schedule, we define *idle time* for parent j any time slot k such that $\min_i t(i,j) < k < \max_i t(i,j)$ and $k \neq t(i,j)$ for any i. In other words, an idle time is a waiting time slot in between meetings and therefore counts as a wasted time.

We define as *school meeting problem* the problem of determining a schedule of minimum makespan that minimizes the total number of idle times over all parents. This way the two objectives of minimizing the makespan and the wasted time of the parents are treated in a lexicographic order. The problem of minimizing the number of idle times within a fixed time is NP-hard, as can be seen by transforming the no-wait open shop problem with 0-1 processing times $O|\text{no-wait}, p_{ij} \in \{0,1\}\,|C_{\max}$, shown to be NP-hard in Gonzalez (1982). Then the school meeting problem is NP-hard. We approach its solution heuristically.

3. Minimizing the Maximum Time for the Meetings

The problem of minimizing the maximum time needed for the meetings is equivalent to a minimum makespan open-shop problem by identifying teachers with machines and parents with jobs. Since the processing times are equal to one, this particular instance of the open shop problem is polynomial and can be easily solved by means of the algorithm by Gonzalez and Sahni (1976) for the open-shop problem with preemption.

The minimum makespan T is given by $T = \max\{\max_j |I_j|, \max_i |J_i|\}$, and an optimal schedule can be computed by solving a sequence of T bipartite matching problems, one for each time slot. In more detail, let $J_i^0 := J_i$ and $I_j^0 := I_j$. Then recursively the k-th matching problem ($k = 1, \ldots, T$) assigns a teacher i to a parent j only if $i \in I_j^{k-1}$. Furthermore, teachers i such that $|J_i^{k-1}| = T - k + 1$ must be assigned to some parent and parents j such that $|I_j^{k-1}| = T - k + 1$ must be assigned to some teacher. These teachers and parents are called *critical* for time k (and remain critical for any subsequent time). Let $M^k \subset I \times J$ be the set of pairs assigned by the k-th matching. Then for all $(i,j) \in M^k$, set $t(i,j) := k$, $I_j^k := I_j^{k-1} \setminus \{i\}$ and $J_i^k := J_i^{k-1} \setminus \{j\}$.

The structure of the above algorithm leaves some room to take into account the lexicographically second objective function of the school meeting problem, that is minimizing the total number of idle times of the parents. Our first strategy consists in solving, at each step, a max weight matching problem instead of a feasible matching problem. To this aim let $s_k(j) \in \{0, 1, 2\}$ be the state of parent j at time k, where the meaning is that $s_k(j) = 0$ if no meeting has been assigned to parent j up to time slot $(k - 1)$ (included), $s_k(j) = 1$ if some, but not all, meetings have been assigned, and $s_k(j) = 2$ if all meetings have been assigned. Then each pair (i, j) receives the weight

$$w_{ij} := \begin{cases} 0 & \text{if } s_k(j) = 0 \\ 1 & \text{if } s_k(j) = 1 \end{cases} \tag{1}$$

in the k-th matching problem (note that if $s_k(j) = 2$ parent j is not considered in the matching problem). Hence, until a parent has not been assigned, his/her meetings receive a zero weight. As soon as a meeting for the parent is assigned, the weight rises to one.

With the cost function (1) the procedure tends to schedule the major part of the meetings in the last slots, when all parents and teachers become critical. This introduces inevitable idle times. In order to overcome this behavior, we have observed that it is useful to modify the cost function by randomly generating the meeting costs so that parents still to be assigned might be encouraged to be assigned a meeting. So we redefine (1) as

$$w_{ij} := \begin{cases} \left.\begin{array}{ll} -1 & \text{with probability } p_1 \\ \ \ 0 & \text{with probability } p_2 \\ \ \ 1 & \text{with probability } 1 - p_1 - p_2 \end{array}\right\} & \text{if } s_k(j) = 0 \\ K\,(1 + W_k(j))^2 & \text{if } s_k(j) = 1 \end{cases} \tag{2}$$

with $W_k(j)$ the number of idle times assigned to parent j during time slots $\{1, \dots, k - 1\}$. The probabilities p_1 and p_2 and the coefficient K must be properly tuned. Hence, until a parent has not been assigned, his/her meetings receive a low weight, strictly lower than one. As soon as a meeting for the parent is assigned, the weight rises to K and then increases quadratically with the number of assigned idle times.

4. Minimizing the Idle Times - Local search LST

The solution found by the procedure described in the previous section may have many idle times. In order to reduce their number, we adopt a large scale neighborhood search approach based on two different types of neighborhood. The first procedure, denoted LST, modifies the schedule by exchanging meetings of a single teacher, while the second procedure, denoted LSP, moves meetings of a single parent. In this section we present the local search LST.

This local search is based on the idea of moving a given meeting to a time slot in which the parent is free. If in the new time slot the teacher is busy with another meeting, this meeting must be moved as well creating recursively a chain of moves. For feasibility, this chain of moves must either be cyclic or end in a free time slot for the teacher. We associate to this chain a cost given by the number of introduced idle times. Although the number of possible moves is exponential, an improving solution in the neighborhood, if any exists, can be found in polynomial time as follows.

Given a schedule with makespan T and a particular teacher i, we define a directed graph $G_i = (N, E_i)$ having T nodes identified with the time slots $1, \dots, T$.

Let B_i be the set of nodes corresponding to the time slots when teacher i is busy with some meeting, i.e. $B_i := \cup_{j \in J_i} t(i, j)$ and $F_i := N \setminus B_i$ be the complement set of nodes when teacher i is free from meetings. Moreover, for each parent j, let H_j be the set of nodes corresponding to the time slots when j is not assigned a meeting, i.e. $H_j := N \setminus \cup_{i \in I_j} t(i, j)$. Note that H_j does not depend on i.

The arcs E_i of the graph G_i are defined as follows. For each $k \in B_i$, let $j(k)$ be the parent meeting i at time k, i.e. $t(i, j(k)) = k$. There is an arc (k, h) for each $h \in H_{j(k)}$. This arc corresponds to moving the meeting of teacher i and parent $j(k)$ from time $k = t(i, j(k))$ to time h. There is no conflict for the parent because the parent is free at time h. There is a conflict for the teacher if $h \in B_i$. But this conflict can be resolved by a subsequent change induced by another arc. Therefore any simple cycle in G_i corresponds to a sequence of changes which eventually lead to a new feasible schedule. Similarly, any simple path terminating in a node $h \in F_i$ corresponds to a feasible sequence of changes. We may extend paths to cycles by adding arcs (h, k) for each $h \in F_i$ and $k \in B_i$.

We assign the following costs to the arcs: the arcs (h, k), $h \in F_i$, $k \in B_i$ receive cost 0; the arcs (k, h), $k \in B_i$, $h \in H_{j(k)}$ receive a cost given by the difference between the idle time number for parent $j(k)$ in the current schedule and in the schedule obtained by moving the meeting from $k = t(i, j(k))$ to time h. Then simple cycles with negative cost correspond to a sequence of changes leading to a schedule with less idle times. Detecting negative cost simple cycles is easy and can be carried out for instance by the Floyd-Warshall algorithm (see Ahuja et al. (1993)). We point out the similarity of this type of search with the one proposed in Dowsland (1998).

The above considerations suggest the following scheme for a large scale neighborhood search: given a schedule, for each teacher i build the graph G_i and detect a negative simple cycle C_i. If there is no such a cycle output a shortest directed cycle. Then select the i's with the most negative cycles and among them randomly select a particular i'. If there are no negative cycles, select randomly any i' with zero cycle cost if any exists. If there are only positive cost cycles stop the procedure, else perform the meeting exchanges indicated by the cycle and continue. Stop the procedure if the number of consecutive zero cost exchanges exceeds a fixed parameter.

5. Minimizing the Idle Times - Local search LSP

In the local search LSP we exchange the roles of teachers and parents. Therefore we move a given meeting to a time slot in which the teacher is free. If in the new time slot the parent is busy with another meeting, this meeting must be moved as well. However, since we count idle times for parents and not for teachers, there is no full symmetry between the two approaches. In order to count the number of idle times introduced or removed by a chain of moves, note that the number of idle times of a single parent changes if and only if his/her total time in the school changes.

The procedure works as follows. Let G_j a graph having nodes corresponding to time slots $1, \ldots, T$ and arcs (k, h) whenever j meets a teacher, let us say i, at time k and i is free at time h. Let $t' := \min_i t(i, j)$ and $t'' := \max_i t(i, j)$. By the above comment, a sequence of switching in the meetings of j changes the number of idle times for j if and only if $t'' - t'$ changes.

In order to find a sequence that reduces the number of idle times of parent j, we solve two shortest path problems in G_j, one from t' and one from t'' to the set H_j of nodes at which parent j is free. The instance with source t' is defined as follows. Let $r \geq 0$ be the number of idle time slots adjacent to t' and $B_j = \cup_i t(i, j)$. We assign to the arcs of G_j the following costs

$$w(k, h) := \begin{cases} -(h - t') & \text{if } k = t',\ h \leq t' + r \\ -(r + 1) & \text{if } k = t',\ h \in H_j,\ t' + r < h < t'' \\ -r + s - 1 & \text{if } k = t',\ h = t'' + s > t'' \\ -r & \text{if } k = t',\ h \in B_j \\ r - (h - t') & \text{if } k \in B_j \setminus \{t'\},\ h \leq t' + r \\ s - 1 & \text{if } k \in B_j \setminus \{t'\},\ h = t'' + s > t'' \\ -1 & \text{if } k \in B_j \setminus \{t'\},\ h \in H_j,\ t' + r < h < t'' \\ 0 & \text{if } k, h \in B_j \setminus \{t'\} \end{cases} \tag{3}$$

Then the cost of a directed path from t' to any node in H_j measures the change in the number of idle times produced by the correponding sequence of moves. This is clearly the case if the path has only one arc (t', h) (first three cases in (3)). If the path has more than one arc, then all arcs different from the first and the last ones correspond to exchanges in between meetings on the same time slots and do not modify the number of idle times. Consistently, they have a null cost (last case) and the change in the idle time number depends only on the first and last arc. The first arc always eliminates r idle times (fourth case). The last arc (k, h) eliminates one more idle time if h is an idle time, while it may introduce idle times if either $h \leq t' + r$ or $h \geq t'' + 1$ (remaining cases). Note that there are no negative cycles and so a shortest path problem is well defined. The instance of the shortest path problem with source t'' can be defined in a similar way.

We define a large scale neighborhood search scheme for LSP as for LST: given a schedule, for each parent j build the graph G_j and compute a shortest path P_j. Then select the j's with the most negative paths and among them randomly select a particular j'. If there are no negative paths, select randomly any j' with zero path cost if any exists. If there are only positive cost paths stop the procedure, else perform the meeting exchanges indicated by the path and continue. Stop the procedure if the number of consecutive zero cost exchanges exceeds a fixed parameter.

In order to extend the neighborhood size, we have also implemented a mixed local search, denoted LSTP, in which we build at each stage both G_j, for all parents j, and G_i for all teachers i and select the best chain of moves among all possibilities.

6. An Example

Let $J = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and $I = \{A, B, C, D, E, F\}$ with meeting requests

$$I_1 = \{A, F\}\,,\ I_2 = \{B, E\}\,,\ I_3 = \{A, B, F\}\,,\ I_4 = \{A, D, E, F\}\,,\ I_5 = \{A, B\}\,,$$

$$I_6 = \{C, E, F\}\,,\ I_7 = \{B, D\}\,,\ I_8 = \{C, E, F\}\,,\ I_9 = \{C, D, E, F\}$$

Then we derive

$$J_A = \{1, 3, 4, 5\}\,,\ J_B = \{2, 3, 5, 7\}\,,\ J_C = \{6, 8, 9\}\,,$$

$$J_D = \{4, 7, 9\}\,,\ J_E = \{2, 4, 6, 8, 9\}\,,\ J_F = \{1, 3, 4, 6, 8, 9\}$$

The minimum makespan of the instance is $T = 6$, with time slots labeled $t_1, \dots, t_6$. By solving the 6 matching problems with a null objective function we obtain the schedule 1 in Fig. 1 (rows refer to parents and columns to time slots, the table entries are the teachers and the $-$ are the idle times), while the schedule 2 has been obtained by introducing the objective function (2).

$$\begin{array}{c} \\ 1\\2\\3\\4\\5\\6\\7\\8\\9 \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{array} \\ \begin{pmatrix} A & F & & & & \\ B & E & & & & \\ F & B & A & & & \\ D & A & - & - & F & E \\ & & B & A & & \\ E & C & - & F & & \\ & D & - & B & & \\ & & E & C & - & F \\ C & - & F & D & E & \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ 1\\2\\3\\4\\5\\6\\7\\8\\9 \end{array} \begin{array}{c} \begin{array}{cccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{array} \\ \begin{pmatrix} F & A & & & & \\ & E & B & & & \\ & F & A & B & & \\ & & E & F & D & A \\ & & & & A & B \\ & & F & C & E & \\ & & & & B & D \\ & & & E & C & F \\ & & C & D & F & E \end{pmatrix} \end{array}$$

schedule 1 schedule 2

Figure 1

As can be seen, the 6 idle times introduced in the first case are eliminated by taking care of them in the objective function. However, idle times do occur with larger instances even if we use (2) and so we need the local search procedures of Sections 4 and 5. We describe in detail the graph G_E for schedule 1 and teacher E, whose meeting sequence is $(6, 2, 8, -, 9, 4)$. Let us denote the six nodes of G_E as $t_1, \ldots, t_6$. Then $B_E = \{t_1, t_2, t_3, t_5, t_6\}$ and $F_E = \{t_4\}$. At time t_1 teacher E meets parent 6. This meeting can be moved to time t_3 with the effect of reducing one idle time, therefore the graph G_E has the arc (t_1, t_3) with cost -1. The meeting can be also moved to t_5 at cost 0 or to t_6 at cost 1. At time t_2 teacher E meets parent 2. The meeting can be moved to t_3, t_4, t_5 or t_6 at cost 1, 2, 3, 4, respectively. Continuing this way, we build the arcs of graph G_E whose costs are reported in the following table.

	t_1	t_2	t_3	t_4	t_5	t_6
t_1	–	–	-1	–	0	1
t_2	–	–	1	2	3	4
t_3	2	1	–	–	-1	–
t_4	0	0	0	–	0	0
t_5	–	-1	–	–	–	1
t_6	–	–	-1	-1	–	–

There are some negative cycles in this graph, for instance $t_1 \to t_3 \to t_5 \to t_6 \to t_4 \to t_1$ with cost -2. Performing the corresponding exchanges leads to the schedule 3 in Fig. 2. Continuing with teacher F, we find the cycle $t_3 \to t_5 \to t_3$ of cost -1 and reduce one more idle time. Then we consider teacher C and by moving the meeting at t_1 to t_3 we reduce two more idle times. Finally, we consider teacher D and move the meeting at t_2 to t_3 obtaining the schedule 4 with no idle times.

$$\begin{array}{c|cccccc} & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \\ \hline 1 & A & F & & & & \\ 2 & B & E & & & & \\ 3 & F & B & A & & & \\ 4 & D & A & - & E & F & \\ 5 & & & B & A & & \\ 6 & & C & E & F & & \\ 7 & & D & - & B & & \\ 8 & & & & C & E & F \\ 9 & C & - & F & D & - & E \end{array}$$

schedule 3

$$\begin{array}{c|cccccc} & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \\ \hline 1 & A & F & & & & \\ 2 & B & E & & & & \\ 3 & F & B & A & & & \\ 4 & D & A & F & E & & \\ 5 & & & B & A & & \\ 6 & & C & E & F & & \\ 7 & & & D & B & & \\ 8 & & & & C & E & F \\ 9 & & & C & D & F & E \end{array}$$

schedule 4

Figure 2

As an example of LSP reconsider schedule 1 in Fig. 1. The graph G_9 related to parent 9 contains two paths of cost -1, precisely $t_1 \to t_6$ and $t_1 \to t_5 \to t_4 \to t_6$. The corresponding sequences of exchanges produce respectively the schedules $(-, -, F, D, E, C)$ and $(-, -, F, E, C, D)$, both having no idle time.

7. A Variant of the Problem

In this section we explain how the previous procedure can be adapted to solve an extension of the school meeting problem in which we allow longer meetings for some previously defined parent-teacher pairs. More precisely, we assume that these meetings last twice as much as the normal ones. We do not allow preemption of the meetings, i.e., a meeting of double length cannot be interrupted and resumed later.

Under the above conditions, even the problem of finding a schedule of minimum makespan becomes NP-hard. Therefore we do not pursue the objective of finding the minimum makespan, rather we consider more important to obtain an initial schedule with no preemption so that the local search procedure can be designed without worrying about the preemption.

Denote by $p_{ij} \in \{1,2\}$ the length of the meeting of teacher i with parent j and by $a_i := \sum_j p_{ij}$, $i \in I$, and $b_j := \sum_i p_{ij}$, $j \in J$, the total time required by the meetings of teacher i and parent j, respectively. Moreover, set $T_{\max} := 2 \max\{\max_i |J_i|, \max_j |I_j|\}$ and $T_{\min} := \max\{\max_i a_i, \max_j b_j\}$. As one can easily verify, the minimum makespan T^* of a schedule that assigns all the required meetings without preemption is bounded by

$$T_{\min} \leq T^* \leq T_{\max} \ ,$$

and a schedule with makespan bounded the same way can be easily obtained (just put $p_{ij} = 2$ for all meetings to obtain a schedule with makespan $T_{\max}$).

In order to solve this variant of the school meeting problem, we modify some aspects of the procedures of Sections 3, 4 and 5. Let us first consider the algorithm of Section 3, which solves a weighted matching problem for each time slot. Since now the makespan T of a feasible solution is not known in advance, even T has to be treated as an output of the solution.

Using the same notation as in Section 3, at the beginning of each step k, three sets J_i^{k-1}, I_j^{k-1} and R^{k-1} are given, where the set R^{k-1} contains all the pairs (i,j) with $p_{ij} = 2$ whose meeting has started at time $k-1$. Let a_i^{k-1} and b_j^{k-1} be the residual times of teacher i and parent j at the end of step $k-1$ and define $T^{k-1} := k - 1 + \max\left\{\max_i |a_i^{k-1}|, \max_j |b_j^{k-1}|\right\}$. Apparently, T^{k-1} is a lower bound on the makespan of any solution coincident with the current partial schedule on the first $k-1$ time slots. We still call critical any parent and teacher having a residual time equal to $T^{k-1} - (k-1)$.

Since preemption is not allowed, all the pairs in R^{k-1} have to be contained in the matching M^k to be determined at the current stage. In order to guarantee that the matching problem admits a feasible solution, no other rigid constraint on the assignment of selected parents and teachers can be imposed. As a consequence, we have to relax the constraints on the critical teachers and/or parents and take care of them in the objective function. Therefore we assign to each pair (i,j) a weight of the form

$$w_{ij} := \begin{cases} K_1 & \text{if } i \text{ or } j \text{ critical} \\ K_2 & \text{else and if } s_k(j) = 1 \\ \left.\begin{array}{ll} -1 & \text{with probability } p_1 \\ 0 & \text{with probability } p_2 \\ 1 & \text{with probability } 1 - p_1 - p_2 \end{array}\right\} & \text{else} \end{cases} \tag{4}$$

where $K_1 \gg K_2$ to take into account in lexicographic order the two objectives of minimizing the makespan and the number of idle times. At the end of the iteration, the sets J_i^k, I_j^k and R^k and the values a_i^k, b_j^k and T^k have to be suitably updated. Note that if at least one critical teacher or parent is not assigned then $T^k = T^{k-1} + 1$. The procedure stops when $a_i^k = 0$ for all i (or equivalently $b_j^k = 0$ for all j). The final makespan is given by the last value T^k.

Also the local search procedures described in Sections 4 and 5 have to be slightly modified to avoid the introduction of any preemption in the meetings requiring two time slots. To this aim, it is sufficient to change the definition of the arc set of the graph G_i related to each teacher i and the graph G_j related to each parent j described in those sections. In the case of G_i, assume that teacher i meets parent j at time t. If the meeting lasts one time slot, then the set of arcs exiting from node t is defined as before. Otherwise, if the meeting takes two time slots, let say t and $t+1$, we add at most two arcs: the arc $(t, t+2)$ if parent j is free at time $t+2$ and the arc $(t+1, t-1)$ if parent j is free at time $t-1$. The construction of the arc set of graph G_j is the same, once the roles of teacher i and parent j are exchanged. The cost of the arcs of the two graphs are defined as the number of idle times of the parents that the corresponding move either adds or eliminates from the schedule. We point out that such a move does not introduce any preemption in the meeting. Obviously, the number of paths in the graph, and consequently the number of moves, is reduced with respect to the case of meetings of equal duration.

8. Computational Results

Let us first consider the main problem with equal time slots. We have tested our procedure on a particular instance given by a local school and also on real size randomly generated data.

Real data to test our procedure are not actually available, because the proposed advance planning of the meetings has never been thought of. However, we have asked some teachers of a local school to provide reasonable data consistent with their past experience. Today each parent cannot meet more than three or four teachers during a meeting session. It is perceived that asking parents to provide a list of prospected teachers to meet would likely produce long lists requiring too much time to carry out all meetings. Therefore it seems more sensible, in a real implementation, to assign an upper bound on the number of teachers a parent can ask to meet.

With this proviso, we have set up the following data, with 29 teachers and 60 parents and a number of required meetings for each parent between 3 and 6. The instance thus obtained has a makespan of 25 time slots. We have run the assignment phase 5 times with parameters $K = 1$, $p_1 = 0.04$, $p_2 = 0.94$, obtaining solutions with 162, 146, 109, 177 and 122 idle times, respectively. Then for each of these solutions we have separately started LST, LSP and LSTP with 20 as the maximum number of consecutive zero improvement cycles. The local search computations are shown in Figs. 3, 4 and 5, where the current idle time number is displayed as a function of the iteration number. We recall that in each iteration a minimum cycle for each teacher has to be computed for LST, a shortest path for each parent for LSP, and both computations have to be carried out for LSTP. The 5 runs for LST, the 5 runs for LSP and the 5 runs for LSTP are shown in Figs. 3, 4 and 5, respectively. The ending idle time numbers for the 5 LST runs are: 4, 0, 3, 3, 2; for the 5 LSP runs are: 22, 15, 10, 21, 23 and for the 5 LSTP runs are: 1, 4, 2, 1, 1. The last of these solutions is shown in Fig. 6.

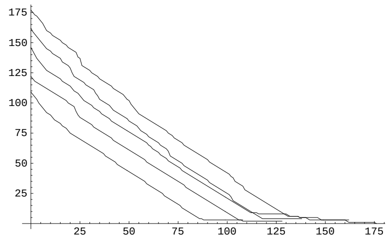

Figure 3 - LST runs

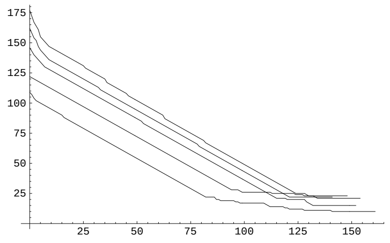

Figure 4 - LSP runs

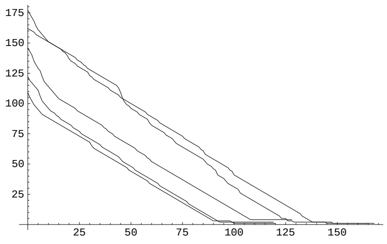

Figure 5 - LSTP runs

As expected, the LST performs much better than the LSP, due to the smaller search space of LSP. On the contrary, the increased neighborhood size offered by the two searches in LSTP does not seem to outperform LST. It is only slightly better on the average.

An interesting feature is that the improvement per iteration is almost constant in all cases, so that we may roughly say that the number of iterations is almost equal to the initial value of the idle times provided by the assignment phase. In this sense the local search performs well independently of the starting solution. However, the number of iterations is affected by the initial solution and therefore it makes sense to tune the parameters of the assignment procedure in order to start with a good solution.

In the random instances we have fixed the number of teachers and parents to 30 and 80 respectively. For each parent the number of required meetings is a random number $r \in [r_1, r_2]$. Then r teachers are randomly selected to meet that parent. We have generated 6 instances. For the instances 1 and 2 we have used $r \in [2, 4]$ (*sparse* instances); for the instances 3 and 4 we have used $r \in [4, 6]$ (*normal* instances) and for the instances 5 and 6 we have used $r \in [4, 8]$ (*dense* instances). For each instance we have run the assignment phase 4 times (with parameters as before) and for each solution of the assignment problem we have separately started LST, LSP and LSTP. The computational results are shown in Tables 1, 2 and 3, where the columns (1) report the instance number (in boldface) and its makespan, the columns (2) report the idle time number at the end of the assignment phase, and, for each local search, the columns (3) and (4) report the final idle time numbers and the iteration numbers respectively. The four rows for each instance refer to the 4 computations.

Table 1 – Sparse instances

		LST		LSP		LSTP	
(1)	(2)	(3)	(4)	(3)	(4)	(3)	(4)
	25	1	39	3	102	2	37
1	13	2	32	5	29	2	31
14	25	2	55	6	67	0	40
	16	2	69	6	59	0	41

		LST		LSP		LSTP	
(1)	(2)	(3)	(4)	(3)	(4)	(3)	(4)
	18	1	58	2	50	1	37
2	18	2	46	7	40	0	25
12	27	1	57	7	65	0	29
	25	3	39	9	34	6	35

Table 2 – Normal instances

		LST		LSP		LSTP	
(1)	(2)	(3)	(4)	(3)	(4)	(3)	(4)
	71	6	71	9	114	8	69
3	58	1	75	23	71	3	63
18	51	8	73	26	58	9	81
	41	1	78	9	101	2	66

		LST		LSP		LSTP	
(1)	(2)	(3)	(4)	(3)	(4)	(3)	(4)
	68	3	118	12	95	2	111
4	54	2	95	10	99	3	98
21	70	1	116	15	130	2	86
	63	1	95	15	86	2	89

Table 3 – Dense instances

		LST		LSP		LSTP	
(1)	(2)	(3)	(4)	(3)	(4)	(3)	(4)
	112	8	136	40	133	5	158
5	131	8	166	38	186	13	110
27	75	7	116	29	89	7	101
	133	8	141	25	186	7	163

		LST		LSP		LSTP	
(1)	(2)	(3)	(4)	(3)	(4)	(3)	(4)
	73	6	140	16	172	12	92
6	110	6	170	4	251	6	167
22	93	10	88	14	184	5	175
	164	14	143	23	254	5	213

As in the previous case the tests show that LST performs much better than LSP. Again, there is no clear winner between LST and LSTP. As a strategy to obtain good solutions, we suggest to run several times both procedures. We have no evident explanation why LSTP is not definitely better than LST.

We have also tested the variant of Section 7 on instances with 15 teachers and 60 parents and a random number of meeting for each parent in $[3, 6]$. The only difference is that each meeting is randomly assigned duration 1 or 2 with probability 0.8 and 0.2, respectively. The matching phase has been carried out with weights given by (4) ($K_1 = 10^4$, $K_2 = 10$, $p_1 = 0.04$, $p_2 = 0.94$). The makespan, the number of idle times at the end of the matching phase and at the end of the modified local search procedure are for the 5 runs 30, 71 and 11; 29, 73 and 21; 30, 76 and 16; 30, 73 and 20; 30, 71 and 11. As anticipated, the local search is not as effective as in the normal case.

$$\left(\begin{array}{rccccccccccccccccccccccccc}
1 & 16 & 26 & 11 & 1 & 21 & 28 \\
2 & & & & & & & & & & 11 & 21 & 1 \\
3 & 28 & 21 & 16 & 11 & 26 \\
4 & & & & & & & & & & & & & & & & & & & & & & 16 & 21 & 28 & 26 \\
5 & & 11 & 1 & 28 & 16 & 21 & 26 \\
6 & & & 28 & 21 & 11 & 1 \\
7 & & & & & & & & & & & & & & & & & 21 & 16 & 11 & 26 & 28 \\
8 & & & & & & & & & 20 & 22 & 11 & 27 & 29 \\
9 & & & & & & & & & & & & & & 22 & 20 & 27 & 29 & 2 \\
10 & & & & & & & & & & 27 & 20 & 22 \\
11 & & & & & & & & & & & & & 11 & 20 & 22 & 2 & 27 & 29 \\
12 & & 2 & 20 & 27 & 22 \\
13 & & & & & & & & & & & & & & & & & & & & & 22 & 2 & 27 & 29 \\
14 & & 12 & 26 & 17 \\
15 & & & & & & & 12 & 17 & 3 & 24 & 28 & 26 \\
16 & 26 & 28 & 17 & 3 & 12 & 24 \\
17 & & & & & & & & & & & & & & & & & & & & & & & 24 & 12 & 17 \\
18 & & & & & & & & & & 3 & 17 & 12 & 26 & 28 & 24 \\
19 & & & & & & & & & & & & 3 & 17 & 24 & 28 & 12 \\
20 & 13 & 4 & 19 \\
21 & & 27 & 13 & 19 & 28 \\
22 & & & & & & & & & & & & 4 & 27 & 13 & 23 & 28 & 19 \\
23 & & & & & & & & & & & & & & & & 4 & 28 & 27 & 23 \\
24 & & & & & & & & & & & & & & 4 & 13 & 19 & 23 & 28 & 27 \\
25 & & & & & & & & & & & & & & & & & & & & & & & 14 & 26 & 29 \\
26 & & & & & & & & & & & & 23 & 14 & 18 \\
27 & & & & 29 & 5 & 23 & 14 \\
28 & & & & & & & 5 & 14 & 26 & 18 & 23 & 29 \\
29 & 29 & 5 & 18 & 26 & 23 & 14 \\
30 & & & & & & & & & & & & & & & & & & & 18 & 5 & 26 & 29 \\
31 & & & & & & & & & & & 14 & 18 & 5 & 23 & 26 & 29 \\
32 & & & & & & & & & & & & & & & & & & & & 6 & 29 & 26 \\
33 & & & 29 & 24 & 13 & 6 & 19 & 26 \\
34 & & & & & & & & & 13 & 19 & 29 \\
35 & & & & & & & & & & & & 19 & 13 & 26 & 29 & 6 & 24 \\
36 & 19 & 29 & 24 & 6 \\
37 & & & & & & & & & & & & & & & & & & & & & 6 & 13 & 29 & 19 & 24 \\
38 & & 24 & 6 & 13 & 29 \\
39 & & 22 & 7 & 16 & 27 & 29 \\
40 & & & 27 & 7 & 14 & 16 & 22 \\
41 & & & & & & & & & & & & & & & & & 16 & 22 & 7 & 29 & 14 & 27 \\
42 & 27 & 7 & 14 \\
43 & & & & & & & & & & & & 14 & 22 & 29 & 27 \\
44 & & & & & & 27 & 16 & 29 & 22 & 14 \\
45 & & 15 & 21 & 20 & 8 & 26 \\
46 & & & & & & & & & & & & & & & 21 & 20 & 15 & 26 & 8 & 28 \\
47 & & & & & & & 20 & 8 & 28 & 15 \\
48 & & & & & & & & & & & 15 & 8 & 20 & 21 \\
49 & & & & & & & 21 & 20 & 15 & 26 & 8 & 28 \\
50 & & & & 9 & - & 12 & 27 & 28 & 25 \\
51 & & & & & & & & & & & & & & & & & & & & & & 17 & 25 & 27 & 28 \\
52 & & & & & & & & & 9 & 12 & 25 & 17 & 28 & 27 \\
53 & & & & & & & & & & & & & & & & & & & & & & & 12 & 17 & 25 \\
54 & & & & 25 & 17 & 9 & 28 & 12 & 27 \\
55 & & & & & & 17 & 25 & 9 & 12 & 28 & 27 \\
56 & & & 10 & 18 & 25 & 15 & 29 & 27 \\
57 & & & & & & & & 18 & 10 & 29 \\
58 & & & & & 15 & 25 & 18 & 10 & 29 \\
59 & & 25 & 15 & 10 \\
60 & & & & & & & & & & & & & & & & & 18 & 15 & 29 & 27
\end{array}\right)$$

Figure 6 – Solution with one idle time

9. References

Ahuja, R.K., Ergun,Ö., Orlin, J.B., Punnen, A.P. (2002), “ A survey of very large-scale neighborhood search techniques”, *Discrete Applied Mathematics*, **123**, 75-102.

Ahuja, R.K., Magnanti, T.L., Orlin, J.B. (1993), *Network flows: theory, algorithms, and applications*, Prentice-Hall, Englewood Cliffs, NJ, USA.

Bartholdi, J.J, McCroan, K.L. (1990), “ Scheduling Interviews for a Job Fair”, *Operations Research*, **38**, 951-960.

Dowsland, K.A. (1998), “Nurse scheduling with tabu search and strategic oscillation”, *European Journal of Operational Research*, **106**, 393-407.

Giaro, K., Kubale, M. (2004), “Compact Scheduling of zero-one time operations in multi-stage system”, *Discrete and Applied Mathematics*, **145**, 95-103.

Gonzalez, T. (1982), “Unit Execution Time Shop Problems”, *Math. Oper. Res.*, **7**, 57–66.

Gonzalez, T., Sahni, S. (1976), “Open Shop Scheduling to Minimize Finish Time”, *Journal of the ACM*, **23**, 665-679.

Hall, N.G., Sriskandarajah, C. (1996), “A Survey of Machine Scheduling Problems with Blocking and No-wait in Process”, *Operations Research*, **44**, 510-525.

Meyers, C., Orlin, J.B. (2006) , “Very Large–Scale Neighborhood Search Techniques in Timetabling Problems”, in *Proceedings of the PATAT 2006 Conference, Brno, Czech Republic*, E. Burke and H. Rudova eds., 36–52.