Search Algorithms for Improving the Pareto Front in a Timetabling Problem with a Solution Network-based Robustness Measure

Can Akkan · Gülcin Ermis

Keywords timetabling · matheuristics · local search · robustness · bi-criteria optimization

1 Introduction

The timetabling problem we address herein was defined by [1], who provided a MIP-based solution approach that uses the *solution pool* feature of CPLEX to obtain a Pareto front.

The timetabling problem requires the assignment of student teams to time slots for presenting their projects taking into account (i) academic diversity of teams, and (ii) homogeneity of types of projects, assigned to each time slot. The course requires three or four teams to present per time slot and all members of the teams in a time slot to be in the audience when other teams present. Diversity of the teams assigned to the time slot is an important concern since many students with different backgrounds take the course. So the problem is to assign teams to time slots that are feasible (in terms of the team members' available times with respect to their course schedules) while aiming to have maximum overall project type homogeneity and team diversity in the time slots used. Taking into account this objective function of the timetabling

This research has been supported by TÜBİTAK Grant 214M661

C. Akkan
Sabanci University, School of Management,
Orhanli, Tuzla, 34956, Istanbul, Turkey Tel.: +90-216-4839685
Fax: +90-216-4839600
E-mail: canakkan@sabanciuniv.edu

G. Ermis
Sabanci University, School of Management,
Orhanli, Tuzla, 34956, Istanbul, Turkey Tel.: +90-216-4832361
Fax: +90-216-4839600
E-mail: gulcinermis@sabanciuniv.edu

problem, denoted by Z, along with a robustness measure, Q, results in a bi-criteria optimization problem. We develop two types of search algorithms (a local search, and a matheuristic) for improving the Pareto front found by the model of [1], which includes at least one of the optimal solutions (usually there are alternative optima) with the objective function value Z^*.

The pool of solutions found for the problem are structured as a network, in which nodes represent solutions and edges are defined by the Hamming distance, $D(a, b)$, between pairs of solutions a and b. $D(a, b)$ gives the number of teams that are not assigned to the same time slots in both solutions. The robustness measure is based on a definition of disruption (learning that the time slot to which an entity had been assigned is no more feasible for that entity) and a predefined heuristic for responding to this disruption (choosing one of the neighbours of the disrupted solution as the new solution). A cutoff value C is assumed, representing the maximum number of teams we are willing to assign to a different time slot in responding to the disruption. Thus, an edge exists between two nodes a and b only if $D(a, b) \leq C$. A solution is robust if the metric, Q given below is small, which happens when either the new solution has a small distance to the disruption (known as solution robustness in the literature), or the new solution does not have a large objective value Z (known as quality robustness), or both.

Measuring the robustness of a given solution, a, is a scenario–based calculation in which scenario i corresponds to team i requesting a time slot change. For scenario i, robustness measure $Q_i(a)$ is calculated as follows:

- Let $\mathcal{C}_i^*(a)$ be the set of neighbours of a where time slot of team i is different from that of solution a.
- If $\mathcal{C}_i^*(a) \neq \emptyset$, let $b_i^* = \arg\min_b\{D(a, b) : b \in \mathcal{C}_i^*(a)\}$. If there is a tie, let b_i^* be the one with the lowest Z. Then, denoting the objective function value of solution b_i^* by Z_i^*, let $Q_i(a) = D(a, b_i^*) \times (Z_i^* - Z^* + 1)$.
- If $\mathcal{C}_i^*(a) = \emptyset$, let $Q_i(a)$ take a large value (a "penalty"), M.

Then, the robustness of solution a, $Q(a)$ is $(1/I)\sum_{i=1}^{I} Q_i(a)$. Assuming each scenario is equally likely, one can consider $Q(a)$ to be representing an expected value.

2 Search Algorithms for Improving the Network

2.1 Local Search Heuristics

Given a solution, d, in the solution pool, these heuristics attempt to create a set of neighbouring solutions to solution d that have a distance at most equal to cutoff C. We call solution d the anchor solution. After generating these neighbours, a new solution is selected as the anchor and this process is repeated until the number of solutions generated exceeds 8000. There are three algorithmic design characteristics, whose combinations result in 32 different local search heuristics. These are:

1. **Neighbour Generation Methods**: In *Limited*, $\mathcal{L}$, only neighbours of d are created by 4 types of local moves. Whereas in *Extended*, $\mathcal{E}$, in addition to the neighbours of d, neighbours of two selected neighbours of d are generated by the local moves in order to find more neighbours of d.
2. **Anchor Selection Set Definitions**: For a given network, selecting the anchor solution is governed by two decisions. First, a subset of the solutions is chosen, which we refer to as the *selection set*. And then a *selection rule* is employed to pick the anchor solution from this selection set. There are two alternative selection set definitions. The first one is *Full set*, $\mathcal{F}$, which is the set of all unvisited solutions. The second one is *Partial set*, $\mathcal{P}$, which is the neighbours generated for the previous anchor solution.
3. **Anchor Selection Rules**: There are seven rules. Three of them, $MIND1$, $MIND2$ and $MIND3$, choose the solution with the smallest *domination count*, differing in their tie-breaking rules. $MAXQ$ and $MINQ$ rules select the solution with maximum and minimum Q value, respectively. $MINZ1$ and $MINZ2$ rules select the solution with minimum Z value, differing in their tie-breaking rules. $IMPD$ rule uses $MIND1$ rule in the first iteration. In the remaining iterations, for each solution p in the selection set, and given the last visited (anchor) solution d, an improvement direction metric, δ_p, is calculated as $\min[(Z(p) - Z(d))/Z(d), (Q(p) - Q(d))/Q(d)]$ and the solution with the minimum δ_p is selected.

2.2 Matheuristic Algorithm

The matheuristic algorithm starts with the initial solution pool, denoted by Π_0, which is obtained by CPLEX using the model presented in [1], which we will denote as MIP_0. This pool, due to its small size, results in a small solution network that yields high Q values. Next, a set of *target* solutions, $\Theta_1 \subset \Pi_0$ is selected. For each solution $\theta \in \Theta_1$, a new integer programming model, MIP_1 is run to obtain neighbouring solutions to θ. The solution pool Π_1, obtained by solving MIP_1 for each solution in Θ_1, is merged with Π_0, to obtain the pool $\Pi = \Pi_0 \cup \Pi_1$ and the associated network. Then, using this new network, for each solution $\theta \in \Theta_1$, we identify the set of teams, $\mathcal{F}(\theta)$, for which no neighbouring solution of θ has a different time slot. Using this information a new set of target solutions $\Theta_2 \subseteq \Theta_1$ is selected. Then, for each $\theta \in \Theta_2$ and for each team $i \in \mathcal{F}(\theta)$, we solve model MIP_2 to obtain the new solution pool, Π_2. This yields the final solution pool $\Pi = \Pi_0 \cup \Pi_1 \cup \Pi_2$. Π is used to create the final solution network and determine the associated Pareto front.

The selection of the target solution set Θ_1 is done by calculating the number of dominating solutions, $ND(v)$, for each solution, v, where the two objectives used are $Z(v)$ and $deg(v)$. Then, $\Theta_1 = \{v : ND(v) \leq D_{max}\}$ (in implementation we chose $D_{max} = 3$).

Formulation MIP_1 is obtained from MIP_0 by removing constraints whose purpose was to prioritize time slots that are likely to yield more robust solutions and adding constraints to ensure that for each new solution b, (i)

$1 \leq D(\theta, b) \leq C$ and (ii) $D(\nu, b) \geq 1$ for each neighbour ν of θ. Finally, formulation MIP_2 is obtained from MIP_1 by removing constraints (ii) (as they are redundant) and adding one constraint that forbids assigning team ϕ to time τ for team $\phi \in \mathcal{F}(\theta)$ and time slot τ to which team ϕ is assigned in the target solution $\theta \in \Theta_2$. Since the objective functions of MIP_1 and MIP_2 are the same as MIP_0, the solution pools created by them are populated by solutions with small Z values.

3 Preliminary Results and Further Work

We use *binary* ϵ-indicator ([2]) to evaluate the quality of fronts found by the heuristics in addition to some summary statistics such as the cardinality of the fronts. Performances are tested on seven semesters' of actual data. Table 1 shows how each local search heuristic performed, using the *binary* ϵ-indicator. For a given heuristic β, *binary* ϵ-indicator was measured with respect to each other heuristic α, denoted by $B_\epsilon(\alpha, \beta)$, such that the larger it is, the better β performs relative to α. Overall performance of β was evaluated by $\overline{B}_\epsilon(\beta)$, which is the average of $B_\epsilon(\alpha, \beta)$ over all α. The results in Table 1 are summary statistics of these $\overline{B}_\epsilon(\beta)$ for the seven semesters' data, showing that $\mathcal{LP} - MIND1$ is the best performer in terms of median performance. The results marked with $*$ are the best performers for each of the semesters. Interestingly, in two of the semesters $MathHeur$ was the best performer. These results suggest that the both the traditional local search algorithms and $MathHeur$ are viable approaches identify good Pareto fronts with robustness as one of the criteria.

References

1. Akkan, C., Külünk, M. E. and Kocas, C., *Finding robust timetables for project presentations of student teams*, European Journal of Operational Research, 249 (2), 560–576, (2016).
2. Zitzler, E. and Thiele, L., *Multiobjective optimization using evolutionary algorithms – A comparative case study*, Parallel Problem Solving from NaturePPSN V, 292–301, (1998).

Table 1 Performance of the Local Search Heuristics Using $\overline{B}_\epsilon(.)$

	Semesters							
	1	2	3	4	5	6	7	$\widetilde{\overline{I}}_\epsilon(.)$
$\mathcal{LF}.MIND1$	2.924	2.329	1.131	10.290	1.047	3.529	1.199	2.329
$\mathcal{LF}.MIND2$	3.545	2.330	1.131	10.290	1.047	3.895	1.199	2.330
$\mathcal{LF}.MIND3$	3.507	2.148	1.131	10.290	1.047	1.139	1.199	1.199
$\mathcal{LF}.MAXQ$	1.041	1.036	1.145	1.211	1.046	1.144	1.065	1.065
$\mathcal{LF}.MINQ$	2.166	2.525	1.085	1.095	5.682	2.887	1.176	2.166
$\mathcal{LF}.MINZ1$	1.419	1.046	1.042	7.697	1.046	1.131	1.649	1.132
$\mathcal{LF}.MINZ2$	1.599	1.193	1.047	7.695	1.046	1.131	1.036	1.132
$\mathcal{LF}.IMPD$	2.954	2.348	1.044	10.307	1.048	3.329	1.198	2.348
$\mathcal{EF}.MIND1$	4.459*	1.532	1.151	10.277	1.047	3.788	1.196	1.532
$\mathcal{EF}.MIND2$	4.405	1.533	1.149	10.277	1.048	3.554	1.195	1.533
$\mathcal{EF}.MIND3$	4.459*	1.533	1.152	10.277	1.047	3.395	1.195	1.533
$\mathcal{EF}.MAXQ$	1.032	1.038	1.145	1.213	1.047	1.036	1.072	1.047
$\mathcal{EF}.MINQ$	2.166	2.321	1.099	1.101	5.682	2.868	1.155	2.166
$\mathcal{EF}.MINZ1$	1.419	1.051	1.078	7.720	1.046	1.055	1.036	1.055
$\mathcal{EF}.MINZ2$	1.469	1.190	1.080	7.695	1.046	1.092	1.036	1.092
$\mathcal{EF}.IMPD$	4.458	1.528	1.139	10.277	1.047	4.387	1.178	1.528
$\mathcal{LP}.MIND1$	2.922	4.473*	1.117	10.286	14.294*	2.940	1.195	2.940
$\mathcal{LP}.MIND2$	3.456	1.533	1.031	10.271	1.047	3.599	1.193	1.533
$\mathcal{LP}.MIND3$	3.462	1.533	1.031	10.271	1.047	1.134	1.195	1.195
$\mathcal{LP}.MAXQ$	1.008	1.035	1.070	1.000	1.006	1.000	1.000	1.006
$\mathcal{LP}.MINQ$	2.166	2.322	1.093	1.007	5.682	2.887	1.178	2.166
$\mathcal{LP}.MINZ1$	1.620	1.133	1.146	3.694	1.046	1.134	1.036	1.134
$\mathcal{LP}.MINZ2$	1.616	1.237	1.172	5.185	1.046	1.134	1.036	1.172
$\mathcal{LP}.IMPD$	2.929	1.221	1.116	10.326*	1.726	2.955	1.036	1.726
$\mathcal{EP}.MIND1$	3.484	2.099	1.151	10.270	1.047	3.529	1.037	2.099
$\mathcal{EP}.MIND2$	4.405	1.532	1.151	10.270	1.047	3.600	1.036	1.532
$\mathcal{EP}.MIND3$	3.484	1.532	1.151	10.270	1.047	3.534	1.037	1.532
$\mathcal{EP}.MAXQ$	1.051	1.050	1.142	3.779	1.113	1.064	1.422	1.113
$\mathcal{EP}.MINQ$	2.271	2.321	1.112	1.101	5.685	2.867	1.173	2.271
$\mathcal{EP}.MINZ1$	1.534	1.088	1.054	7.712	1.048	1.125	1.085	1.088
$\mathcal{EP}.MINZ2$	2.952	1.124	1.100	7.676	1.048	1.131	1.036	1.124
$\mathcal{EP}.IMPD$	4.459	2.162	1.146	10.312	1.048	4.389*	1.178	2.162
$MathHeur$	2.317	1.570	3.218*	2.880	1.819	2.265	1.720*	2.265