Bi-Criteria Simulated Annealing Algorithms for the Robust University Course Timetabling Problem

Ayla Gülcü · Can Akkan

Abstract A bi-criteria version of the curriculum-based university timetabling problem of ITC-2007 is solved using a multi-objective simulated annealing (MOSA) algorithm that identifies an approximation to the optimal Pareto front. The two criteria are the penalty function as defined in ITC-2007 and a robustness function. The robustness function assumes one disruption occurs in the form of a period of an event (lecture) becoming infeasible for that event. The parameters of the MOSA algorithm are set using the Iterated F-Race algorithm and then its performance is tested against a hybrid MOGA algorithm developed by the authors. The results show that MOSA provides better approximation fronts than the hybrid MOGA.

Keywords University course timetabling · Robustness · Bi-criteria optimization · Multi-objective simulated annealing

1 Introduction

For the curriculum-based university course timetabling problem defined in ITC-2007 (see [6]), we assume that a disruption can occur in the form of a period to which an event of a course has been assigned, ceasing to be feasible for that event. If such a disruption occurs, the timetable has to be updated,

This research has been supported by TÜBİTAK Grant No: 217M475

A. Gülcü
Fatih Sultan Mehmet University, Dept. of Computer Science, İstanbul, Turkey.
Tel.: +90-212-5218100
Fax: +90-212-3698164
E-mail: fagulcu@fsm.edu.tr

C. Akkan
Sabancı University, School of Management, İstanbul, Turkey.
Tel.: +90-216-4839685
Fax: +90-216-4839600
E-mail: canakkan@sabanciuniv.edu

in essence re-optimized, while ensuring that the changes to the rest of the timetable are limited. Hence, a timetable is said to be *robust* if, when disrupted, its feasibility can be restored without significantly lowering its quality in terms of the objective function while keeping it relatively stable. We formulate the problem of identifying a robust timetable as a bi-criteria optimization problem where one objective is the quality of the solution measured as a function of the violated soft constraints, that is the penalty function denoted by P, and the second one is a function that measures the robustness of the timetable, denoted by R. The problem is formulated as one of finding an approximation to the optimal Pareto front defined by P and R.

This problem was first defined in [1] and solved using a hybrid Multi-objective Genetic Algorithm (MOGA), which makes use of Hill Climbing and Simulated Annealing algorithms in addition to the standard Genetic Algorithm approach. Here we propose a Multi-objective Simulated Annealing (MOSA) algorithm that outperforms the hybrid MOGA on the ITC-2007 data set.

2 The robustness measure

Calculation of the robustness measure R requires finding, for each lecture E, the move that yields the minimum incremental penalty $r(E)$ to restore feasibility if that lecture is disrupted. We assume there are three type of moves allowed for this re-optimization problem, and the move that yields $r(E)$ is the minimum cost one among all the feasible moves of these types: (1) moving only the disrupted lecture to an empty room in a feasible time slot (called the simple move), (2) swapping the disrupted lecture with another lecture (called the swap move) and (3) the Kempe chain move of [4] which enables a lecture to be moved more flexibly to a new time slot.

When a swap or a Kempe chain move is made, in addition to the change in the penalty costs of the moved lectures, a fixed cost is added. This fixed cost is calculated as the average per lecture penalty, P_{ave}, in a set of 1200 randomly generated feasible solutions for the given instance, denoted by $\mathcal{S}$. These solutions are the same solutions generated in the initial populations (each of size 40) of 30 runs of the multi-objective Genetic Algorithm (MOGA) of [1]. Thus, $P_{ave} = \frac{1}{|\mathcal{S}||\mathcal{E}|}\sum_{S\in\mathcal{S}} P(S)$, where $\mathcal{E}$ is the set of lectures.

The robustness of solution S is calculated as $R(S) = \frac{1}{|\mathcal{E}|}\sum_{E\in\mathcal{E}} r^{+}(E)$, where $r^{+}(E) = \max(r(E), 0)$. The search for $r^{+}(E)$ stops if a move with $r^{+}(E) = 0$ is found, or all feasible moves for that lecture have been evaluated. As the current robustness measure is computationally expensive, we employed a heuristic measure to estimate the robustness of a given solution. The heuristic robustness measure takes a small subset of lectures into account, by taking a fixed fraction of lectures, denoted by *PercentLec*, from each curriculum in increasing order of their saturation degrees ([5]). Saturation degree of a lecture is the number of valid periods, in terms of satisfying the hard constraints, in the timetable. This subset is formed for each instance only once before the start of the algorithm, and it remains the same throughout the running of the algorithm. This ensures that each curriculum is represented in the sample and for each curriculum, the most difficult events in terms of time availabilities are selected. This approach, rather than randomly selecting events, gave strongly positively correlated robustness results with the real robustness measure where all lectures have been evaluated.

3 The Multi-objective Simulated Annealing (MOSA) Algorithm

We designed a two-phase MOSA, where, in the first phase, the algorithm works as a SA algorithm for minimizing the penalty function P, and then in the second phase it works as a MOSA to find the best approximation front for the objectives P and R.

In both of these phases we use a real time initial temperature selection strategy that employs a short "burn-in" period in which worsening moves are accepted with a predefined probability. The total number of iterations is determined so that the algorithm takes twice the competition time limit (as we try to solve a two-objective problem). Furthermore, in order to not to violate the time constraints of the timetabling competition, we restricted the time spent on Stage 1 to be at most equal to the competition time limit (the (*Phase 1 CPU Sec*)/(*Total CPU Sec*) parameter in Table 1). To control how much effort the MOSA algorithm spends on improving R versus P, we limited the local moves from a given solution to have P values that are at most a certain multiple of the best penalty found in phase 1 (the *Max* (P/*Phase_1_P**) parameter in Table 1).

The main difference between a MOSA algorithm and a single-objective SA algorithm is in the design of the acceptance rule and the maintenance of an archive of solutions that yield the front (for a survey of MOSA and SA algorithms see [11]). Let $z_k(X)$, for $k \in \{1,2\}$, be the value of the k^{th} objective for solution X. To decide whether a move from solution X to X' is accepted, let $\Delta Z = (\Delta z_k, k \in \{1,2\})$ with $\Delta z_k = z_k(X') - z_k(X)$. Then, assuming all criteria are to be minimized, if $\Delta z_k \leq 0\ \forall k$, X' is always accepted. Otherwise, probabilistic acceptance rules can be used. We experimented with three types of rules. **Probability scalarizing** rules first calculate the acceptance probability of the move from X to X' for each objective individually, then a decision rule is applied in order to aggregate these probabilities, giving a relative weight λ_k to each objective k. The acceptance probability with respect to the k^{th} objective, π_k, is computed as shown in Equation 1.

$$\pi_k = \begin{cases} exp\left(-\frac{\Delta z_k}{T}\right), & \text{if } \Delta z_k > 0 \\ 1, & \text{if } \Delta z_k \leq 0 \end{cases} \tag{1}$$

In the **criterion scalarization** rules, multiple objective values are combined in order to create a single value using a weight vector, λ_k, for each criterion k. The most widely used scalarizing approach is the weighted sum method which is shown in Equation 2, where $S(\Delta Z)$ represents the scalarized objective function value. The acceptance probability is then computed as shown in Equation 3 (see [8]).

$$S(\Delta Z) = \lambda \Delta z_1 + (1-\lambda)\Delta z_2, \text{ where } 0 < \lambda < 1. \tag{2}$$

$$p = \begin{cases} \exp\left(-\frac{S(\Delta Z)}{T}\right), & \text{if } S(\Delta Z) > 0 \\ 1, & \text{if } S(\Delta Z) \leq 0 \end{cases} \tag{3}$$

In the **Pareto Domination** based approaches the acceptance probability of a given solution is computed by comparing it with the set of potentially Pareto-optimal feasible solutions. Based on these three types of rules we tested eight acceptance rules:

1. **PSmin**: For the probability scalarizing with minimum rule, the aggregate acceptance probability, p, is computed as $p = \min_{k=1,..,K}(\pi_k)^{\lambda_k}$.
2. **PSmax**: For the probability scalarizing with maximum rule the aggregate acceptance probability, p, is computed as $p = \max_{k=1,..,K}(\pi_k)^{\lambda_k}$.
3. **PSprod**:For the probability scalarizing with product rule the aggregate acceptance probability, p, is computed $p = \prod_{k=1}^{K}(\pi_k)^{\lambda_k}$.
4. **CS$_{\lambda=0.5}$**: The criterion scalarizing approach with $\lambda = 0.5$.
5. **CS$_{\lambda=0.7}$**: The criterion scalarizing approach with $\lambda = 0.7$.
6. **CS$_{\lambda=0.3}$**: The criterion scalarizing approach with $\lambda = 0.3$.
7. **Suman**([10]): Let $|F(X)|$ be equal the number of solutions in the potentially Pareto-optimal solutions dominating solution X plus 1. Then, $\Delta S = |F(X)| - |F(X^{'})|$ and the aggregate acceptance probability, p, is computed as $p = \min\{1, \exp(-\frac{\Delta S}{T})\}$.
8. **Smith** ([9]): Letting $\widetilde{F}$ denote the union of the current potentially Pareto optimal set F, the current solution X and the new solution X', modifies the Suman rule by setting $\Delta S = 1/|\widetilde{F}| \left(|F(X)| - |F(X^{'})|\right)$.

We used an *Iterated F-Race* experiment (see [3]) to identify the best-performing settings of the design parameters given in Table 1. Eight parameters, each having 3, 3, 3, 4, 8, 3, 2 and 4 levels, respectively, yields a total of 20736 configurations. Clearly, a full factorial experiment would not be viable. By using Iterated F-Race we were able to set the total computational budget, B, to just 440 runs. This resulted in the number of configurations to be tested to be 13, 11, 10, 9, 8 and the numbers of instances used to test these configurations to be 7, 8, 9, 10, 11, for iterations 1 through 6, respectively.

To rank the configurations, the racing algorithm requires the cost function to be a scalar number, which is referred to as *utility* in [7]. Since we are dealing with a multi-objective problem, the utility function should be able to quantify the quality of a Pareto front. We defined the utility function as weighted sum of *generational distance*, *spread*, *spacing* and *minimum distance to the minimum-penalty point*. The first three metrics are also used to evaluate the overall performance of our MOSA algorithm; their exact definitions are given in [1]. All these metrics are defined so that they take values in $[0, 1]$,

Table 1 Parameter settings in the Iterated F-Race experiment

Parameter	Settings			
$\frac{Phase\ 1\ CPU\ Sec}{Total\ CPU\ Sec}$	(1) 0.3	(2) 0.4	(3) 0.5	
Phase 1 T_0 Prob.	(1) 0.5	(2) 0.7	(3) 0.9	
Phase 2 T_0 Prob.	(1) 0.3	(2) 0.5	(3) 0.7	
$Max\frac{P}{Phase\_1\_P^*}$	(1) 1	(2) 1.5	(3) 2	(4) 2.5
Acceptance rule	(1) PSmin	(2) PSmax	(3) PSprod	(4) $CS_{\lambda=0.5}$
	(5) $CS_{\lambda=0.7}$	(6) $CS_{\lambda=0.3}$	(7) Suman	(8) Smith
$PercentLec$	(1) 10	(2) 20	(3) 30	
Kempe move	(1) yes	(2) no		
Phase 2 cooling rate	(1) 0.99	(2) 0.95	(3) 0.9	(4) 0.85

and all the metrics except *spread* decrease with improving solution quality. The fourth one, *Minimum distance to minimum-penalty point*, is included to provide bias towards solutions with smaller P values, since ultimately we seek robust solutions with small P values assuming a decision maker would be willing to choose a solution with better robustness only if that that solutions does not have an excessively inferior penalty value. Then, the utility of each front i obtained by parameter configuration t, U_i^t, is computed by as $0.5 * D_{min-p}(F_i^t) + 0.25 * GD(F_i^t) + 0.15 * (1 - S(F_i^t)) + 0.1 * S_p(F_i^t)$ (these are the same weights used in [1]).

MOSA parameter settings corresponding to the F-Race winning configuration are, *Phase 1 CPU Sec/Total CPU Sec* = 0.4; *Phase 1* T_0 *Prob.* = 0.7; *Phase 2* T_0 *Prob.* = 0.7; $MaxP/Phase\_1\_P^*$ = 2.5; *Acceptance rule* = 8; $PercentLec$ = 30; *Kempe move* = yes, and *Phase 2 cooling rate* = 0.95. In addition to these parameters, two parameters were fixed as a design decision. First, we used one constructive heuristic to create an initial feasible solution. Second, for the first phase of the MOSA, we set the cooling rate cr to 0.99, because [2] determined that for these competition instances setting a specific value for the cooling rate, at least within the investigated intervals, was essentially irrelevant to the performance, and they fixed the cooling rate to 0.99 in their experiments.

4 Results

The performance of MOSA is tested on the dataset for the CB-CTP track of ITC-2007 ([6]), which is comprised of 21 instances. Table 2 presents some statistics on the P and R values obtained in the 30 runs done using MOGA and MOSA for each instance. Excluding the trivial instance 11, among the remaining 20 instances, MOSA improved both $\overline{P^*}$ and $\min(P_i^*)$ for 19 instances.

MOGA yielded a better $\overline{P^*}$ only for instances 1 and 18, and yielded a better $\min(P_i^*)$ only for instance 18. Furthermore, these penalty performances are competitive with respect to the best known penalty values in the literature. On the other hand, $\overline{K}$ statistics show that the fronts obtained by MOGA span an extremely wide range of P values, whereas those of MOSA are concentrated close to the P^* of the front. We believe one can argue that a decision maker would not be willing to deteriorate the penalty of a solution so excessively in order to improve robustness, thus in the remaining comparative analysis of the results we decided to focus on the Pareto fronts within a window of P values defined by the range $[P^{best}, 5 \times P^{best}]$, where P^{best} equals to the minimum of P^* found by MOSA and P^* found by MOGA, for each instance.

Table 2 Some P and R statistics on the fronts

	MOGA				MOSA			
	$\overline{P^*}$	$\overline{R^*}$	$\min(P_i^*)$	$\overline{K}$	$\overline{P^*}$	$\overline{R^*}$	$\min(P_i^*)$	$\overline{K}$
1	5.00	0.05	5	480.8	5.50	1.51	5	13.2
2	79.37	0.66	63	68.1	73.43	1.38	55	4.1
3	100.24	0.22	84	54.5	93.83	0.68	76	3.8
4	47.77	0.06	42	95.1	45.43	0.73	41	4.7
5	377.44	1.72	333	22.1	349.23	2.69	316	2.8
6	74.60	0.32	66	71.1	68.67	1.67	53	4.0
7	45.74	0.32	33	109.8	37.5	1.86	26	5.0
8	55.47	0.08	47	72.1	50.80	0.75	40	4.2
9	118.90	0.09	109	38.3	114.2	0.53	105	3.3
10	38.90	0.22	29	116.7	35.50	1.27	25	5.9
11	0	0	0	-	0	0	0	-
12	370.10	0.73	346	11.57	357.77	1.52	339	2.7
13	88.40	0.04	80	71.1	82.87	0.84	73	3.6
14	67.64	0.12	58	56.9	66.90	0.86	57	3.8
15	100.3	0.21	89	56.6	94.80	0.68	82	3.8
16	59.07	0.26	49	71.9	56.03	1.14	37	4.2
17	97.90	0.21	88	51.4	91.37	1.02	80	3.6
18	90.07	0.01	78	22.4	94.17	0.49	80	3.4
19	80.44	0.29	72	55.4	78.30	0.83	68	3.5
20	65.87	0.34	55	111.7	60.30	2.79	42	4.3
21	130.54	0.23	116	38.4	119.2	0.68	105	3.4

Note: $K_i = P_i^+/P_i^*$, where P_i^+ and P_i^* are the maximum and minimum penalty found in the i^{th} run

Table 3 provides statistics on the average values of several metrics for the fronts found in this window of P values (the averages of the 30 runs for each instance), where all metrics except for $\overline{NB_\epsilon}$ are calculated using normalized P and R values. $\overline{C}$ denotes the average number of solutions in the fronts obtained in 30 runs for each instance. The average number of unique objective vectors in the fronts is denoted by $\overline{V}$. $\overline{S} \in (0, 1]$ denotes the average spread of the fronts, such that spread equals 1 if a front includes both of the extreme solutions in the aggregate front A^*. A^* is constructed using all 30 fronts found by MOSA and all 30 fronts found MOGA, for each instance. Thus, for each instance A^* is the best Pareto front known. $\overline{Sp} \in [0,1]$ denotes the average spacing value of the final fronts. The more the solutions are uniformly spaced in a given front, the smaller the spacing metric becomes. $\overline{NB_\epsilon}$ uses the *better* relationship based on the binary ϵ-indicator (see [12]). For the MOSA statistics, for each front, F_i, found by the i^{th} run of MOSA, the number of times F_i is better than the fronts obtained by MOGA, $NB_{\epsilon,i}$, is calculated (that is, each such F_i is compared with 30 fronts). This is repeated for the 30 runs of MOSA for that instance, yielding 30 $NB_{\epsilon,i}$ measurements and their average is denoted by $\overline{NB_\epsilon}$. $\overline{GD}$ is the average of the generational distances between each front and A^*, thus the smaller GD is, the better the front is. Clearly, the larger $\overline{NB_\epsilon}$ values are associated with better performance. Finally, $\overline{HV}$ denotes the average hypervolume, so that larger values represent better fronts. The results in Table 3 show that for only the spacing metric the two algorithms yield a similar performance and for all other metrics MOSA is significantly better. We performed one-sided Wilcoxon signed rank test for each performance metric, in which the null hypothesis is MOGA yields a better performance than MOSA. For all but the spacing metric, this null hypothesis is rejected, verifying that MOSA is better than MOGA. We are currently working on a version of this algorithm that can handle multiple disruptions of the same type and hope to present at least some preliminary results for that in the conference as well.

Table 3 Performance comparison for solutions with P in $[P^{best}, 5 \times P^{best}]$ for all instances except the 11^{th}

	MOGA				MOSA				
	Min	Ave	Med	Max	Min	Ave	Med	Max	p-value
$\overline{C}$	2.54	7.59	5.54	18.80	16.74	25.24	25.49	36.77	9.70e-6
$\overline{V}$	2.47	7.52	5.39	18.70	10.27	23.38	23.74	35.34	9.70e-6
$\overline{S}$	0.16	0.52	0.53	0.85	0.60	0.74	0.74	0.81	0.0004
$\overline{Sp}$	0.06	0.10	0.10	0.16	0.07	0.11	0.10	0.20	0.895
$\overline{NB_\epsilon}$	0.00	0.04	0.00	0.20	0.57	4.10	3.55	10.99	9.70e-6
$\overline{GD}$	0.06	0.09	0.09	0.13	0.01	0.03	0.02	0.05	9.36e-6
$\overline{HV}$	0.04	0.13	0.12	0.27	0.08	0.22	0.22	0.44	9.59e-6

References

1. Akkan, C., Gülcü, A.: A bi-criteria hybrid Genetic Algorithm with robustness objective for the course timetabling problem. Computers and Operations Research **90**, 22–32 (2018)
2. Bellio, R., Ceschia, S., Di Gaspero, L., Schaerf, A., Urli, T.: Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. Computers and Operations Research **65**, 83–92 (2016)
3. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-Race and Iterated F-Race: An Overview, pp. 311–336. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
4. Burke, E., Eckersley, A., McCollum, B., Petrovic, S., Qu, R.: Hybrid variable neighbourhood approaches to university exam timetabling. Tech. Rep. NOTTCS-TR-2006-2, University of Nottingham, School of CSiT (2006)

5. Cheong, C.Y., Tan, K.C., Veeravalli, B.: A multi-objective evolutionary algorithm for examination timetabling. J. of Scheduling **12**(2), 121–146 (2009)
6. Di Gaspero, L., McCollum, B., Schaerf, A.: The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3). Tech. Rep. QUBIEEETechITC2007CurriculumCTTv1.0, Queens University, Belfast, United Kingdom (2007)
7. Eiben A. E.and Smit, S.K.: Evolutionary Algorithm Parameters and Methods to Tune Them, pp. 15–36. Springer Berlin Heidelberg (2012)
8. Serafini, P.: Simulated Annealing for Multi Objective Optimization Problems, pp. 283–292. Springer New York, New York, NY (1994)
9. Smith, K.I., Everson, R.M., Fieldsend, J.E., Murphy, C., Misra, R.: Dominance-based multiobjective simulated annealing. IEEE Transactions on Evolutionary Computation **12**(3), 323–342 (2008)
10. Suman, B.: Simulated annealing-based multiobjective algorithms and their application for system reliability. Engineering Optimization **35**(4), 391–416 (2003)
11. Suman, B., Kumar, P.: A survey of simulated annealing as a tool for single and multi-objective optimization. Journal of the operational research society **57**(10), 1143–1160 (2006)
12. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation **7**(2), 117–132 (2003)