Empirical Analysis of Tabu Search for the Lexicographic Optimization of the Examination Timetabling Problem

Luis Paquete and Thomas Stützle

Intellektik,Technische Universität Darmstadt,
Alexanderstr. 10, 64283 Darmstadt, Germany
{lpaquete,tom}@intellektik.informatik.tu-darmstadt.de
www.intellektik.informatik.tu-darmstadt.de/{~lpaquete,~tom}

1 The Examination Timetabling Problem

An examination timetable of an educational institution is a mapping $t : E \rightarrow T$, where E is a set of examinations and T is a set of time slots. The Examination Timetabling Problem ETP(E, T, f) is the problem of finding an optimal mapping $s_{opt} \in S$, where S is the set of all possible mappings from E to T, such that, given an cost function $f : S \rightarrow R^+$, $f(s_{opt}) \leq f(s), \forall s \in S$. Other formulations also include the assignment of rooms.

The cost function is usually related to the number of violations of several types of constraints. Since most real life problems are over-constrained, one possibility to attack this problem is to order the constraints in a hierarchy according to their importance to the institution.

In the simplest case, one can distinguish between hard and soft constraints. While the former are required to hold for any feasible solution, the latter express a solution's quality by their degree of violation. A typical hard constraint is to forbid a temporal overlap of a pair of examinations for a same student. Typical examples of a soft constraint is a minimum temporal distance between a pair of examinations for a same student or precedence between examinations (see [2] for an extensive list of possible constraints found in real life problems).

These constraint types have to be taken into account when solving the problem by a stochastic search algorithm, and several possibilities have been proposed to accommodate them. One approach is to assign each objective f_i a weight λ_i that expresses the importance of the i-th objective and then to minimize

$$f(s) = \sum_{i=1}^{n} \lambda_i f_i(s) \tag{1}$$

which corresponds to the *weighted sum* of the objectives f_i [5]. By defining violations of constraints as objectives to be minimized, such a formulation can be used to attack the ETP. In this case, the highest weight is assigned to the objective related to the satisfaction of the hard constraints. However, this approach is only useful if the user is able to express his preferences by setting weights.

If setting explicit weights is not possible, a way out can be a priority order of constraints given by the user. This approach is commonly known as *lexicographic optimization* [13] or *multi-phase* approach [14, 15]. Like in the previous approach it is assumed that the user is able to express his preferences in a hierarchical way, in which the hard constraints are assigned the highest priority ranks. Finally, in the case of lack of preference information given *a priori*, the ETP should be regarded as a typical multiobjective problem with the goal of finding Pareto solutions [1, 11].

In this study our interest lies in the lexicographic formulation of the Examination Timetabling Problem and its solution by a Tabu Search algorithm, which is described in the next section.

2 Tabu Search Implementation

When adapting a local search algorithm to a lexicographic optimization problem, a question arises about how to direct the search process, given the priority order information on the objectives. We considered two distinct strategies, Lex-tie and Lex-seq. The former compares solutions by the objective function value of the higher priority objective, and in case of a tie, the solutions are compared using the next lower priority objective. The Lex-seq strategy consists in finding sequentially regions of the search space which satisfy the constraints associated with the decreasing priority order of objectives, that is, first a solution satisfying *all* highest priority constraint is sought, next a solution satisfying all second priority constraints is sought, etc., subject to the fact that no constraints of a higher priority level are violated anymore. It should be stressed that solving the first objective corresponds to solving a Graph Coloring Problem (GCP). We can also consider this approach as an algorithm in which constraints are incrementally added during the search process.

Due to the similarity with the GCP [3], we adapted a Tabu Search algorithm that gave good results for several hard GCP instances [8, 12]. The algorithm uses a 1-opt neighborhood that at each step changes the time slot assignment of exactly one examination. At each step, all pairs of examinations and time slots (e_j, t) are considered and a move is applied that maximally reduces the number of constraint violations. The neighborhood size is reduced by considering moves that only affect examinations that are currently involved in a violation of the constraints considered by the Lex-tie and Lex-seq strategies.

Since the Tabu Search needs a tabu list of forbidden moves to prevent cycling, its length must be defined. In our case, we defined it by $Random(a) + \alpha c$, where c is the number of constraint violations according to the strategy chosen: all constraints in case of Lex-tie, and only the constraints associated to the objective currently being minimized, in case of Lex-seq. $Random(a)$ is random integer between 0 and a.

Similar speed-up techniques for the neighborhood evaluation as proposed in [6] are used: By defining n objectives, the implementation defines a three-dimensional table of size $n \times |E| \times |T|$ where each entry $\Delta(n_i, e_j, t_k)$ stores the effect on the i-th constraint level incurred by changing the time slot of examination j to time slot k. Each time a move is performed, only the part of each table that is affected by the move is updated. In Lex-tie, the initialization has complexity $O(n \times |E|^2 \times |T|)$, and each update has a

worst case complexity of $O(n \times |E| \times |T|)$. In Lex-seq, when the i-th objective reaches null cost, the $(i+1)$-st dimension of the table corresponding to the following objective is initialized, which has a complexity $O(|E|^2 \times |T|)$. When i objectives are solved, each update has a complexity of $O((i+1) \times |E| \times |T|)$. Thus, the initialization and the update only consider the current objective to be minimized and the ones already optimized.

3 Experimental Results

3.1 Average and Peak Performance

To compare the Lex-tie and Lex-seq strategies, we used benchmark instances available at ftp://ftp.mie.utoronto.ca/pub/carter/testprob. These benchmark instances are real life Examination Timetabling Problems that have already been used in [2, 4, 7, 15]. A hard constraint corresponding to no temporal overlapping between a pair of examinations of the same student was considered. The number of violations of this constraint was defined as the highest priority objective to minimize.

Analogously to [4], the soft constraints were defined according to the temporal distance between the same pair of examinations. The minimum temporal distance allowed was 6 time slots. Every pair with lower value was considered as a constraint violation. Following [10], the same values for penalizing these violations according to the temporal distance between the two assignments were used. Thus, a lower priority objective was formulated as a weighted sum of the violations of these constraints.

In some preliminary experiments, we first identified good parameter settings for the Tabu Search algorithm. We found that good performance was achieved in most of the instances by the setting of $A = 10$ and $\alpha \in \{4, 8, 12, 16\}$. For reference, also $\alpha = 1$ was considered. Table 1 presents the best results obtained by the proposed algorithm, after running 25 experiments with 100000 iterations for each instance and for each α. The number of time slots for each instance was the same as in [7]. The Avg column presents the average cost function value of the lowest priority objective divided by the number of students as proposed in [4]. These averages only refer to the runs that satisfied the hard constraints. The number of successful runs, that is those that satisfied all hard constraints, is presented in the column Suc. The Min column presents the minimum objective value found in the successful runs. The best average results considering the maximum number of successful runs and minimum values found for each instance and for each approach, are represented in bold face.

The results seem to reveal that the best value for α increases with the size of the instance in case of the Lex-seq. It is also possible to observe the similarity in the cost function value of the two approaches for small instances, provided a good setting for α is used. However, for large instances, the performance differences are enormous. The results of Lex-tie indicate less efficiency with instances with more than 300 examinations. The Lex-seq performed better in most of the instances when considering different values for α, meaning that it is less sensitive to this parameter.

Table 2 compares our results with the ones obtained by Gaspero and Schaerf [7], and Carter *et al.* [4] when considering peak performance. The former presented a Tabu Search algorithm with a weighted sum approach with shifting penalties to prevent local

Table 1. Experimental results

				Lex-tie			Lex-seq		
Instances	Exams	Time slots	α	Suc	Avg	Min	Suc	Avg	Min
HEC-S-92	80	18	1	25	**12.0**	**11.2**	25	**12.4**	**11.7**
			4	25	12.7	12.2	25	13.0	12.5
			8	25	13.6	12.4	25	13.1	12.0
			12	9	13.8	12.7	25	13.0	11.9
			16	6	13.6	12.1	25	13.2	12.2
STA-F-83	138	13	1	23	163.7	159.3	0	-	-
			4	25	**159.3**	**158.1**	1	161.9	161.9
			8	25	161.5	158.7	25	**168.7**	161.3
			12	18	162.2	159.2	25	169.4	161.9
			16	17	162.4	159.0	25	169.8	**161.2**
YOR-F-83	180	21	1	22	**41.5**	**40.0**	25	**41.7**	**38.9**
			4	13	44.2	41.7	25	43.8	41.3
			8	5	44.3	42.0	25	44.2	42.6
			12	0	-	-	25	44.7	41.9
			16	0	-	-	25	44.7	41.1
UTE-S-92	184	10	1	5	30.8	29.0	25	31.5	29.6
			4	25	**29.4**	**27.8**	25	**30.5**	**28.7**
			8	18	31.2	29.1	25	31.3	28.9
			12	20	31.3	28.5	25	31.6	28.8
			16	20	31.4	29.6	25	32.0	29.1
EAR-F-83	189	24	1	0	-	-	6	48.6	46.0
			4	17	**42.0**	**38.9**	12	44.3	**40.5**
			8	0	-	-	15	47.1	44.4
			12	0	-	-	19	45.8	42.9
			16	0	-	-	23	**45.8**	42.5
TRE-S-92	261	23	1	0	-	-	25	10.8	10.0
			4	6	**9.9**	**9.6**	25	**10.2**	**9.3**
			8	20	10.3	9.8	25	10.4	9.7
			12	0	0	0	25	10.5	9.6
			16	0	0	0	25	10.7	10.1
LSE-F-91	381	18	1	0	-	-	4	16.1	14.7
			4	0	-	-	14	14.5	**13.2**
			8	0	-	-	20	14.9	13.4
			12	1	**13.7**	**13.7**	21	15.5	14.2
			16	0	-	-	24	**15.5**	14.3
KFU-S-93	461	20	1	0	-	-	8	19.1	17.9
			4	0	-	-	22	18.6	16.7
			8	0	-	-	25	**18.3**	**16.5**
			12	0	-	-	25	**18.3**	16.9
			16	0	-	-	25	18.6	16.9

Table 2. Comparison of results

	Lex-tie		Lex-seq		[7]		[4]
Instances	Avg(suc)	Min	Avg(suc)	Min	Avg	Min	Min
HEC-S-92	**12.0** (25)	11.2	12.4 (25)	11.7	12.6	12.4	**10.8**
STA-F-83	**159.3** (25)	**158.1**	168.7 (25)	161.2	166.8	160.8	161.5
YOR-F-83	41.5 (22)	40.0	**41.7** (25)	**38.9**	42.1	41.0	41.7
UTE-S-92	**29.4** (25)	27.8	30.5 (25)	28.7	31.3	29.0	**25.8**
EAR-F-83	42.0 (17)	38.9	**45.8** (23)	40.5	46.7	45.7	**36.4**
TRE-S-92	9.9 (20)	9.6	**10.2** (25)	**9.3**	10.5	10.0	9.6
LSE-F-91	13.7 (1)	13.7	**15.5** (24)	13.2	15.9	15.5	**10.5**
KFU-S-93	- (0)	-	**18.3** (25)	16.5	19.5	18.0	**14.0**

optima. The size of the tabu list was defined randomly in a certain interval during the search. Carter *et al.* used a combination of a backtracking strategy with a saturation degree sorting heuristic. The average cost function values obtained by our approaches were lower than ones in [7]. The minimum objective value obtained was lower than in [4, 7] for three instances.

3.2 Run-time Distributions

To understand the reason for such large differences in the results obtained for some instances, we analyzed the algorithm behavior during the minimization of the higher priority objective, since both approaches deal with it in a different way. One reasonable way is to compute Run-Time Distributions (RTDs) as proposed in [9], which give the empirical probability of finding a solution (satisfaction of the hard constraints) as a function of the run-time (number of iterations).

Figures 1, 2 and 3 show the RTDs obtained for several values of α by both approaches for the instances UTE-S-92, TRE-S-92 and EAR-F-83, respectively. One observation is the dependence of the Lex-tie performance on the difficulty in satisfying the hard constraints. For the instances where the Lex-seq approach satisfies the hard constraints very quickly (in this case the RTDs are positioned mainly to the left with high success rates), that is on instances UTE-S-92 and TRE-S-92, also Lex-tie can do so with similar performance. However, if the Lex-seq approach has some difficulty in satisfying the hard constraints (RTDs skewed to the right), Lex-tie presents much worse performance. It should be stressed that, even with a general worse performance regarding the satisfaction of hard constraints, Lex-tie was able to obtain better average performance considering the minimization of the soft constraints in case it could satisfy the hard constraints (e.g., comparing average performance and RTDs on instance EAR-F-83). However, this observation seems to be limited to small instances.

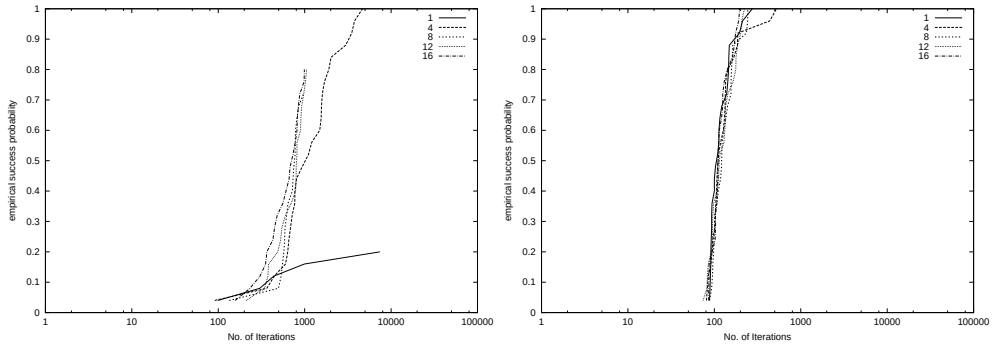

Fig. 1. RTDs of Lex-tie (left) and Lex-seq (right) on UTE-S-92 with $\alpha \in \{1, 4, 8, 12, 16\}$

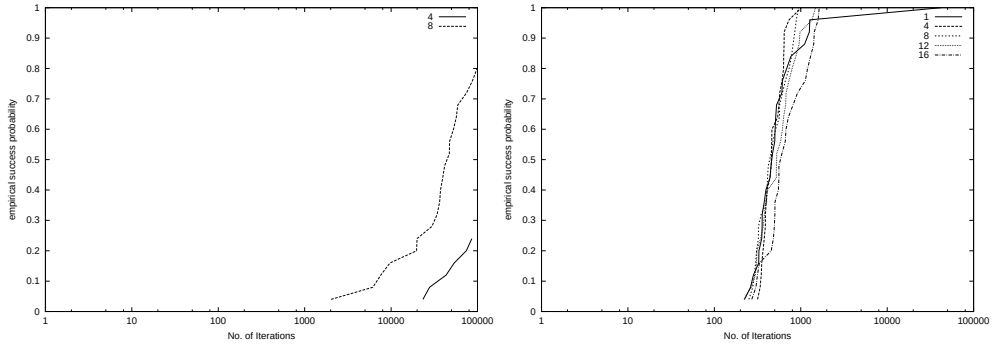

Fig. 2. RTDs of Lex-tie (left) and Lex-seq (right) on TRE-S-92 with $\alpha \in \{1, 4, 8, 12, 16\}$

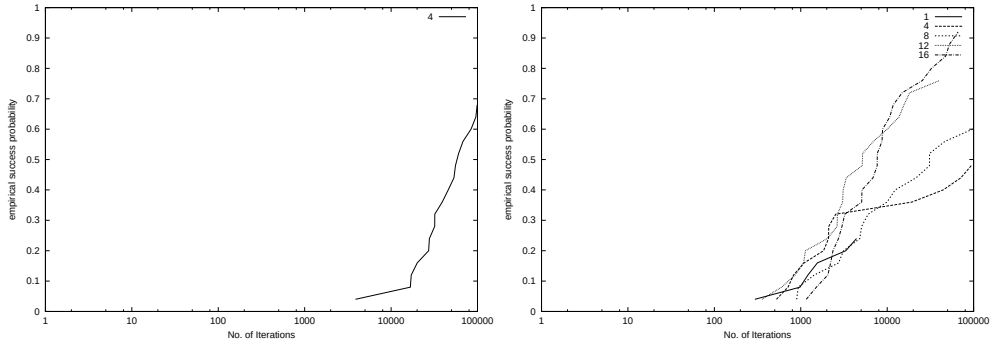

Fig. 3. RTDs of Lex-tie (left) and Lex-seq (right) on EAR-F-83 with $\alpha \in \{1, 4, 8, 12, 16\}$

4 Conclusions

Two main conclusions can be drawn from the two strategies. Firstly, the level of diversification given by α must increase with the size of the instance in the Lex-seq approach.

Secondly, the Lex-tie strategy presents a strong performance deterioration as the hard constraints become more difficult to satisfy. However, the good results obtained by Lex-tie for some instances show that a promising future approach could be combining both strategies by switching between them during the run of the algorithm.

Acknowledgments We would like to thank Marco Chiarandini for valuable discussions on the topic of this research. This work was supported by the "Metaheuristics Network", a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

References

1. E. Burke, T. Bykov, and S. Petrovic. A multicriteria approach to examination timetabling. In E. Burke and W. Erben, editors, *The Practice and Theory of Automated Timetabling III*, Lecture Notes in Computer Science 2079, pages 118–131. Springer-Verlag, 2001.
2. E. Burke, D. Elliman, P. Ford, and R.Weare. Examination timetabling in british universities - a survey. In E. Burke and P. Ross, editors, *The Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science 1153, pages 76–90. Springer-Verlag, 1996.
3. M. Carter. A survey of practical applications of examination timetabling problem algorithms. *Operations Research*, 34:193–202, 1986.
4. M. Carter, G. Laporte, and S. Lee. Examination timetabling: Algorithms strategies and applications. *Journal of Operations Research Society*, 74:373–383, 1996.
5. D. Corne, H.-L. Fang, and C. Mellis. Solving the modular exam scheduling problem with genetic algorithm. Technical Report 622, Department of Artificial Intelligence, University of Edinburgh, 1993.
6. C. Fleurent and J. Ferland. Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, 63:437–464, 1996.
7. L. Gaspero and A. Schaerf. Tabu search tecnhiques for the examination timetabling. In E. Burke and W. Erben, editors, *The Practice and Theory of Automated Timetabling III*, Lecture Notes in Computer Science 2079, pages 104–117. Springer-Verlag, 2001.
8. J. Hao and R. Dorne. Empirical studies of heuristic local search for constraint solving. In *Proceedings of Constraint Programming (CP-96)*, Lecture Notes in Computer Science, pages 194–208. Springer-Verlag, 1996.
9. H.H. Hoos and T. Stützle. Evaluating Las Vegas algorithms, pitfalls and remedies. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 238–245, 1998.
10. G. Laporte and S. Desroches. Examination timetabling by computer. *Computers & Operations Research*, 11:361–372, 1984.
11. L. Paquete and C. Fonseca. A study of examination timetabling with multiobjective evolutionary algorithm. In *Proceedings of the 4th Metaheuristics International Conference (MIC 2001)*, pages 149–154, 2001.
12. L. Paquete and T. Stützle. Experimental investigation of iterated local search for coloring graphs. In S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. Raidl, editors, *Applications of Evolutionary Computing - EvoWorkshops 2002*, Lecture Notes in Computer Science 2279, pages 122–131. Springer-Verlag, 2002.
13. R. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application*. John Wiley & Sons, 1986.

14. J. Thompson and K. Dowsland. Variants of simulated annealing for the examination timetabling problem. In G. Laporte and L. Hosman, editors, *Annals of Operations Research*, volume 63, pages 105–128. Baltzer Science Publishers, 1996.
15. G. White and B. Xie. Examination timetabling and tabu search with longer-term memory. In E. Burke and W. Erben, editors, *The Practice and Theory of Automated Timetabling III*, Lecture Notes in Computer Science 2079, pages 85–103. Springer-Verlag, 2001.