Finding the Answer by Looking in the Wrong Place

Elias Oliveira[1]* and Barbara M. Smith[2]

[1] School of Computing, University of Leeds, Leeds LS2 9JT, U.K.
elias@comp.leeds.ac.uk
[2] School of Computing & Mathematics, University of Huddersfield, Huddersfield HD1 3DH, U.K.
b.m.smith@hud.ac.uk

Abstract. We can model a constrained optimization problem as a constraint satisfaction problem with a variable representing the objective, and repeatedly solve the problem with increasingly tight bounds on the objective variable, until no solution can be found. In practice, this is often difficult because the bounds on the objective variable do not propagate well to the other variables of the problem.
However, we may be able to find a variable related to the objective which behaves much better as an objective variable, so that a bound imposed on this variable is immediately propagated to the search variables and reduces their domains. We propose to use such a variable as a surrogate objective, and show how we can benefit from the improved propagation while still guaranteeing that the right optimal solution, in terms of the true objective, can be found.
We show that in a train scheduling problem this scheme gives much better results than using the true objective directly.

1 Introduction

An optimization problem in constraint programming can be modelled by a constraint satisfaction problem (CSP) with an objective variable, representing the quantity to be optimized. Any solution to the CSP should assign a value to this variable. The optimal solution can be found by repeatedly solving the CSP and using the value of the objective variable in each solution as a new bound (upper or lower, depending on whether we are minimizing or maximizing) on the value of this variable in any future solution. When the bound on this variable becomes so tight that there is no solution to the CSP, the last solution found is known to be the optimal solution. Alternatively, since each solution found is better than the previous one, the process can be terminated whenever finding a new solution becomes too difficult, in the reasonable hope that the current solution is not too far from optimal.

* The first author is a PhD student partially funded by the Brazilian government sponsoring agency, CAPES, under Grant BEX–1162/96-9.

A common source of difficulty with this optimization scheme is that the updated bound on the objective variable given by a new solution may not immediately yield tighter bounds on the search variables. Hence, finding a new, improved, solution may take a long time. When the optimal solution has been found, it may not be possible, without a great deal of search, to detect that the current bound on the objective cannot be met and so prove optimality.

In such a case, it may be possible to find a variable related to the objective which does behave much better as an objective variable, *i.e.* a bound imposed on this variable is immediately propagated to the search variables and reduces their domains. If this were our objective, we should be able to find an optimal solution much more quickly. On the face of it, this is not helpful, since optimizing the wrong objective will find the wrong optimal solution. However, we propose to use such a variable as a surrogate objective, and show how we can benefit from the improved propagation while still guaranteeing that the right optimal solution, in terms of the true objective, can be found.

The problem we are using as an example in this paper consists of scheduling a sequence of tasks grouped into jobs. The tasks in a job must be performed consecutively, with no breaks between tasks, and processed on a number of machines in a sequence specified for each job. Each machine can only process one task at a time. In reality, the problem is one of scheduling trains on a single-track railway, but has been remodelled as a job-shop scheduling problem [5]. Each job represents a journey in the original problem, and each task represents the movement of a train along a stretch of track between signals.

A desired schedule with the release date of each job is given. However, this schedule may have conflicts: more than one task scheduled on a machine at the same time. The problem is then to (re)schedule the tasks to eliminate the conflicts in such a way that the sum of the delays is minimized. The delay of a job is measured by the difference between the actual completion time of completion of the job and its desired completion time.

This paper is structured as follows. In Section 2 we describe how our surrogate objective is engineered for our problem example. In Section 3 we present a comparison of the results using the surrogate objective against a branch-and-bound algorithm for the original objective. We conclude the paper in Section 4 by discussing the potential for other applications.

2 Translating Objectives

The objective is to find a schedule which minimizes the total tardiness, or total delay, of the set of jobs. For each job J_i, T_i is a variable representing its delay. If $\mathcal{T} = \{T_1, T_2, \ldots, T_{|\mathcal{J}|}\}$ is the set of tardiness variables for the jobs $J_i \in \mathcal{J}$, the objective is to minimize $S = \sum_{T_i \in T} T_i$.

We also consider the maximum tardiness, $T_{max} = \max\{T_1, T_2, \ldots, T_{|\mathcal{J}|}\}$. Amongst the set of possible schedules, it can be expected that there is a positive correlation between the values of T_{max} and those of S, although the solution which minimizes T_{max} does not necessarily minimize S. Minimizing T_{max} is likely to be easier than minimizing S, because any new upper bound on T_{max} immediately gives a new upper bound on each individual T_i, whereas a new upper bound on S does not usually lead to any immediate reduction in the upper bounds of the T_i variables.

In the light of this, we propose to use T_{max} as a surrogate objective and minimize it rather than directly minimizing the total delay. However, when a new solution is found, the total delay in that solution will provide a new upper bound on S. Consequently, when an 'optimal' solution is found, *i.e.* when we cannot find a solution with smaller maximum delay than the incumbent solution, it will not give the true minimum of T_{max}, but the minimum subject to the current bound on S. At the same time, the value of S in this solution will likely be less than the value in the solution that truly minimizes T_{max}.

We start from an initial solution found by a heuristic described in [4], in which the value of T_{max} is $\overline{T}_{max}$, and minimize T_{max} by binary search within the range $[0, \overline{T}_{max}]$, subject to the upper bound on total delay S, which is updated whenever a new solution is found.

When no further solution can be found, we know that there is no schedule which has both smaller total delay and smaller maximum delay. Hence, if this solution does not minimize S, the optimal solution must have a larger value of T_{max}. Let T'_{max} be the value of T_{max} in this solution.

In the second phase of the optimization, we *maximize* T_{max}, again subject to the continually revised upper bound on S. Let T''_{max} be the value of T_{max} in the last solution found in this maximization phase. Then we know that there is no schedule with both a larger value of T_{max} and a smaller value of S. If this is not the optimal solution (*i.e.* the solution that minimizes S), the optimal solution must have a smaller value of T_{max}.

At this point, after first minimizing T_{max} and then maximizing it, we know that the optimal solution for S has T_{max} between T'_{max} and T''_{max}, and S is no greater than its value in the last solution found.

We now repeat the minimization phase, searching between T'_{max} and T''_{max}, and then the maximization phase, and repeat those phases alternately until the domain of T_{max} is reduced to a single value: this must yield the solution with minimum total delay.

Figure 1 shows the first minimization and maximization phases for an example scheduling problem. The initial solution has $S = 5068$ and $T_{max} = 1573$. Successive solutions in the minimization phase have total delay 5056, 4939, 4518 and maximum delay 1492, 793, 647 respectively. Since no solution can be found with smaller total delay and smaller maximum delay, we know that the optimal solution has total delay 4518 or less, and maximum delay 647 or more. We then maximize T_{max}, while imposing increasingly tight bounds on the total delay, and find solutions with total delay 2993, 2922, 2875 and values of T_{max} 765, 856, 1082 respectively. Again, we know that the optimal solution has $S \leq 2875$ and 647 $\leq T_{max} \leq$ 1082. Repeated alternation of the minimization and maximization phases will eventually find the optimal solution, which has total delay 1524 and maximum delay 876.

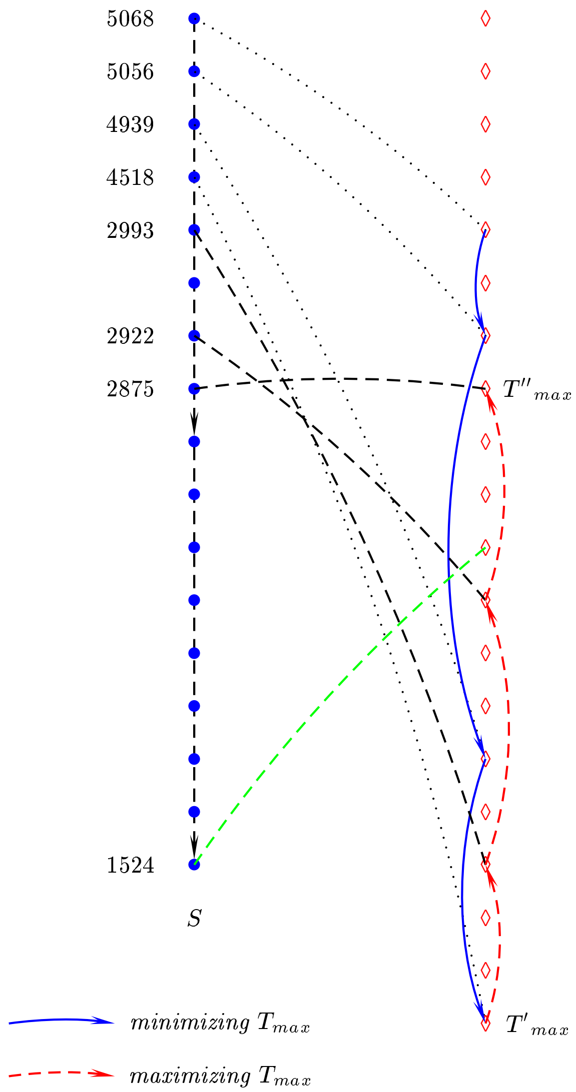

Fig. 1. A parallel diagram linking the values of the surrogate objective, T_{max}, and the true objective, S. (Not to scale.)

3 Results

We compare the performance of our proposed technique against a branch-and-bound algorithm described in [4]. This branch-and-bound algorithm uses only bounds on the total tardiness objective to prune the search space. The algorithm repeatedly resolves the conflicts in the desired schedule, until none remain. In order to resolve a conflict between tasks of different jobs, the release dates of one or more jobs need to be re-timed, because each job's tasks must be performed one immediately after another (see Section 1). The initial upper bound on the total delay is given by the initial solution found by a heuristic described in [4]. Subsequently, whenever a solution is found its cost updates the upper bound for the next solution, and the algorithm searches for further improved solutions until it reaches the optimal solution.

Both the branch-and-bound and the algorithm for the surrogate objective strategy presented in this paper are implemented using ILOG Scheduler [2, 3], a constraint programming framework implemented in C++ on top of ILOG Solver. ILOG Scheduler provides the user with features to model class of resources and tasks for solving problems that involve scheduling tasks over a period of time.

The dataset used in our experiments, gathered from Higgins [1], consists of 21 single-track railway scheduling problems. The problem sizes range from 7 jobs with 12 tasks to 30 jobs with 32 tasks on the two largest problems, 60 and 61.

The experiments were performed on a networked PC Pentium III. The results of applying the approach proposed in this paper and the branch-and-bound strategy described above are compared in Table 1. Column # shows the problem number. The *First Solution* columns show the time (in seconds) to find the first feasible solution to a problem and its respective delay: this solution is yielded by the heuristic described in [4]. The *B&B* columns show the total time for the branch-and-bound algorithm to find and prove the optimality of a solution, and the delay in the best solution found, if the search did not terminate. The time given does not include the time to find the initial solution. The *Surrogate* columns show similar results for the surrogate objective scheme based on T_{max}. The final column shows the delay in the optimal solution, where this is known. For all but the last two problems, both schemes found the optimal solution and proved its optimality.

We set a limit time of 15 hours for each problem, both for the surrogate scheme and the branch-and-bound algorithm. Where the search did not terminate within that time, we give the delay in the best solution found. Note that the branch-and-bound algorithm did not improve on the initial solution to problems 60 and 61 within 15 hours running time.

For the smaller problems (20 to 46) the branch-and-bound algorithm can find an optimal solution and prove optimality quite quickly: for these problems there is not a great advantage in using the surrogate scheme. However, for the larger problem the surrogate scheme is much better. With problems 50 and 51, the time to find and prove the optimal solution is greatly reduced. Problems 60 and 61 are difficult for both algorithms, and neither can find the optimal

Table 1. A comparison between the surrogate scheme and B&B.

#	First Solution Time	First Solution Delay	B & B Time	B & B Delay	Surrogate Time	Surrogate Delay	Optimal Delay
20	0.05	2993	1.96		0.73		1524
21	0.02	723	0.14		0.25		375
22	0.02	206	0.02		0.04		206
23	0.03	427	0.10		0.14		339
24	0.02	627	0.10		0.15		448
25	0.02	1173	0.99		0.07		305
26	0.02	474	0.09		0.10		390
27	0.01	320	0.03		0.06		212
28	0.03	441	0.04		0.08		348
29	0.02	633	0.26		0.11		357
40	0.03	357	0.06		0.15		329
41	0.03	766	0.90		1.06		766
42	0.05	1514	6.33		0.42		527
43	0.03	1007	0.91		1.22		766
44	0.04	950	1.52		0.39		562
45	0.04	993	2.44		0.25		358
46	0.06	1574	4.81		1.67		801
50	0.60	6275	29748		1087		1015
51	0.33	2755	2320		511		970
60	2556	4502	54000	(4502)	54000	(1884)	
61	651	5767	54000	(5767)	54000	(5756)	

solution. However, the solutions found by the surrogate scheme are much better than those found by branch-and-bound.

4 Conclusion

In this paper we present a scheme which uses a surrogate objective when the true objective is difficult to optimize. We show that by using each new solution to bound the true objective, and alternately minimizing and maximizing the surrogate objective, we can guarantee that when the process terminates the optimal solution in terms of the true objective will have been found. We compare this scheme with one that optimizes the true objective directly, on a class of train scheduling problems. For the most difficult problems, using the maximum delay as an aid to minimizing the total delay gives much better results than directly minimizing the total delay. When minimizing the maximum delay, each new solution gives a bound on the delay of each task which aids the search for a better solution.

Although we present results for only one class of problem, we believe that the technique presented in this paper could well be applied to other classes of problem whenever dealing with an inconvenient objective.

References

1. Higgins, A. *Optimisation of Train Schedules to Minimise Transit Time and Maximise Reliability.* PhD thesis, School of Mathematics/School of Civil Engineering, Queensland University of Technology Australia, 1997.
2. ILOG S.A. Ilog *Scheduler Reference Manual, 4.0 edition 1997.*
3. ILOG S.A. Ilog *Scheduler User Guide Manual, 4.0 edition 1997.*
4. Oliveira, E. *Solving Single-Track Railway Scheduling Problem Using Constraint Programming.* PhD thesis, School of Computing, University of Leeds, September 2001.
5. Oliveira, E. and Smith, Barbara M. A Job-Shop Scheduling Model for the Single-Track Railway Scheduling Problem. pages 145–160. 19th Workshop of the UK Planning and Scheduling Special Interest Group – PlanSIG, 2000. Also available at `http:/www.comp.leeds.ac.uk/sacm`.