Closing the Open Shop: Contradicting Conventional Wisdom

Diarmuid Grimes[1], Emmanuel Hebrard[1], and Arnaud Malapert[2]

[1] Cork Constraint Computation Centre & University College Cork, Ireland
{d.grimes|e.hebrard}@4c.ucc.ie
[2] EMN/LINA UMR CNRS 6241, CIRRELT
arnaud.malapert@emn.fr

Abstract. This paper describes a new approach for solving disjunctive temporal problems such as the open shop and job shop scheduling domains. Much previous research in systematic search approaches for these problems has focused on developing problem specific constraint propagators and ordering heuristics. Indeed, the common belief is that many of these problems are too difficult to solve without such domain specific models. We introduce a simple constraint model that combines a generic adaptive heuristic with naive propagation, and show that it often outperforms state-of-the-art solvers for both open shop and job shop problems.

1 Introduction

It is usually accepted that the most efficient methods for solving *Open shop* and *Job shop* scheduling problems are local search algorithms, such as tabu search for job shop [16, 17] and particle swarm optimization for open shop [20]. However, constraint programming often remains the solution of choice. It is indeed relatively competitive [4, 23] whilst providing a more flexible approach. For instance, one can add domain-specific constraints without entailing major design revisions. Moreover, it is commonly believed that the most efficient CP models are those based on strong inference methods, such as *Edge Finding* [7, 18], and specific search strategies, such as *Texture* [9].

In this paper we build on the results of [1] which showed that for open shop problems a constraint programming approach with strong inference and domain specific search strategies outperforms the best local search algorithms. Here, we introduce a constraint model combining simple propagation methods with the generic *weighted degree* heuristic [5] and empirically show that the complex inference methods and search strategies can, surprisingly, be advantageously replaced by this naive model.

An $n \times m$ open shop problem (OSP) involves n jobs and m machines. A job is composed of m tasks, each associated with a duration and a machine. Each machine maps to exactly one task in each job, and can only run a single task at a time. The objective is to minimise the *makespan* M, that is, the total duration to run all tasks. A job shop problem (JSP) is identical except the order of the tasks within each job is fixed.

Our approach relies on a standard model and generic variable ordering and restart policy. First, each pair of conflicting tasks, whether because they belong to the same job, or share a machine, are associated through a *disjunctive* constraint, with a Boolean

variable standing for their relative ordering. Following the standard search procedure for this class of problems, the search space can thus be restricted to the partial orders on tasks. Second, the choice of the next (Boolean) variable to branch on is made by combining the current domain sizes of the associated two tasks with the weighted degree of the corresponding ternary disjunctive constraint. Third, we use restarts, together with a certain amount of randomization and nogood recording from restarts [13].

We demonstrate that this simple approach outperforms more sophisticated constraint models using state of the art heuristics and filtering algorithms implemented in Choco [8] and Ilog Solver, both on OSPs and JSPs. We believe that the weighted degree heuristic, within this model, is extremely effective at identifying the contentious pairs of tasks. In Section 2 we first describe state of the art constraint models and strategies for open shop and job shop scheduling problems. Next, we describe a lighter model as well as the search strategies that we use to empirically support our claims. In Section 3, we present an experimental comparison of our model with state-of-the-art solvers for open shop and job shop scheduling problems.

2 Constraint Models

Here, we describe the two models that we compared in our experiments on the open shop and job shop benchmarks. The first model is the accepted state of the art, using strong inference (global unary resource constraints with the Edge Finding filtering algorithm) as well as specific search heuristics. The second is our lighter and simpler approach, relying on ternary reified disjunctive constraints, and a variable heuristic largely based on the generic weighted degree heuristic [5]. We shall refer to the former as the "heavy" model, and to the latter as the "light" model throughout.

Edge Finding + Profile, "Heavy" Model: In this model, a global filtering algorithm is used for each unary resource. Let T denote a set of tasks sharing an unary resource and Ω denote a subset of T . We consider the three following propagation rules:

Not First/Not Last: This rule determines if the task t_i cannot be scheduled after or before a set of tasks Ω. In that case, at least one task from the set must be scheduled after (resp. before). The domain of task t_i can be updated accordingly.

Detectable Precedence: If a precedence $t_i \prec t_j$ can be discovered by comparing t_i and t_j's time bounds, then their domains can be updated with respect to all the predecessors or successors.

Edge Finding: This filtering technique determines that some tasks must be executed first or last in a set $\Omega \subseteq T$. It is the counterpart of the first rule.

Search Strategy: The branching scheme is that proposed in [3] and denoted *Profile*. We select a critical pair of tasks sharing the same unary resource and impose an ordering. This heuristic, based on the probabilistic profile of the tasks, determines the most constrained resources and tasks. At each node, the resource and the time point with the maximum contention are identified, then a pair of tasks that rely most on this resource at this time point are selected (it is also ensured that the two tasks are not already connected by a path of temporal constraints).

Once the pair of tasks has been chosen, the order of the precedence has to be decided. For that purpose, we retain one of the three randomized value ordering heuristics from the same paper: centroid. The centroid is a real deterministic function of the domain and is computed for the two critical tasks. The centroid of a task is the point that divides its probabilistic profile equally. We commit the sequence which preserves the ordering of the centroids of the two tasks. If the centroids are at the same position, a random ordering is chosen. (For a more detailed discussion on filtering techniques for disjunctive scheduling problems we would point the reader to [2].)

Simple Disjunction + Weighted Degree, "Light" Model: The starting time of each task t_i is represented by a variable $t_i \in [0..M - d_i]$. Next, for every pair of unordered tasks t_i, t_j sharing a job or a machine we introduce a Boolean variable b_{ij} standing for the ordering between t_i and t_j. A value of 0 for b_{ij} means that task t_i should precede task t_j, whilst a value of 1 stands for the opposite ordering. The variables t_i, t_j and b_{ij} are linked by the following constraint, on which Bounds Consistency (BC) is maintained:

$$b_{ij} = \begin{cases} 0 \Leftrightarrow t_i + d_i \leq t_j \\ 1 \Leftrightarrow t_j + d_j \leq t_i \end{cases}$$

For n jobs and m machines, this model therefore involves $nm(m + n - 2)/2$ Boolean variables for OSPs, $nm(m + n - 2)/4$ for JSPs, and as many disjunctive constraints.

Search Strategy: Instead of searching by assigning a starting time to a single value on the left branches, and forbidding this value on the right branches, it is common to branch on *precedences*. In the heavy model, an unresolved pair of tasks t_i, t_j is selected and the constraint $t_i + d_i \leq t_j$ is posted on the left branch whilst $t_j + d_j \leq t_i$ is posted on the right branch. In our model, branching on the Boolean variables precisely simulates this strategy and thus significantly reduces the search space. Indeed, it has been observed (for instance in [15]) that the existence of a partial ordering of the tasks (compatible with start times and durations, and such that its projection on any job or machine is a total order) is equivalent to the existence of a solution. In other words, if we successfully assign all Boolean variables, the existence of a solution is guaranteed.

We use the weighted degree heuristic [5], which chooses the variable maximising the total weight of neighbouring constraints, initialised to its degree. A constraint's weight is incremented by one each time the constraint causes a failure during search. We show in Section 3, that the weighted degree heuristic is very efficient in this context. It is important to stress that the behaviour of this heuristic is dependent on the modelling choices. Indeed, two different, yet logically equivalent, sets of constraints may distribute the weights differently. In this model, every constraint involves one and only one search variable. Moreover, the relative light weight of the model allows the search engine to explore much more nodes, thus learning weights quicker.

However, at the start of the search, this heuristic is completely uninformed since every Boolean variable has the same degree (i.e. 1). We use the domain size of the two tasks t_i, t_j associated to every disjunct b_{ij} to inform the variable selection method until the weighted degrees effectively kick in. We denote $w(ij)$ the number of times the search failed while propagating the constraint on t_i, t_j and b_{ij}. We pick the variable minimising the sum of the residual time windows of the two tasks' starting times, divided by the weighted degree: $(max(t_i) + max(t_j) - min(t_i) - min(t_j) + 2)/w(ij)$

3 Experimental Section

All experiments reported in this paper were run on an Intel Xeon 2.66GHz machine with 12GB of ram on Fedora 9. Each algorithm run on a problem had an overall time limit of 3600s. Unless otherwise stated, values were chosen lexically, ties were broken randomly and nogoods were recorded from restarts.

We first provide evidence that, contrary to popular belief, the domain-specific model and heuristics are unnecessary for solving these problems. For OSPs, we compare our model with the heavy model which was recently shown to be the state-of-the-art on these benchmarks, matching the best metaheuristics on the Taillard benchmarks, and outperforming both exact and approximate methods on the other two benchmarks [1]. We implemented our algorithm in Mistral [11] and, for better comparison on the OSPs, in Choco. The code and parameters used for the heavy model are those used in [1].

Next, on JSPs, we compare the same light model implemented in Mistral, with the heavy model implemented in Ilog Scheduler (algorithm denoted "randomized restart" in [4]). Once again, we got the code for our comparison from the author and used the same parameters as were used in that paper.

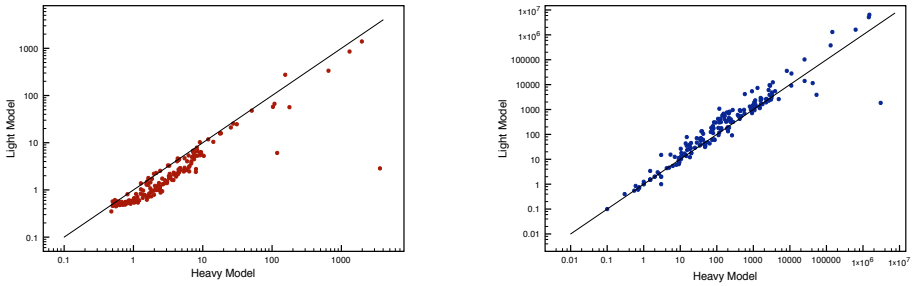
3.1 Open shop scheduling

We used three widely studied sets of instances: Brucker [6], Gueret & Prins [10], and Taillard [21]. The problems range from size 3x3 to 20x20, with 192 instances overall. For all experiments, a sample of 20 runs with different seeds was performed.

The Choco models use a simple randomised constructive heuristic detailed in [1], and referred to as "CROSH", to calculate a good initial upper bound, followed by branch and bound search. The restarting strategy used for the light model (both Choco and Mistral) was geometric [22], with a base of 256 failures, and a multiplicative factor of 1.3. The heavy model uses the Luby restarting sequence [14] order 3 with a scale factor tuned to the problem dimensions. We also ran the most effective Mistral model which only differs from the Choco light model in that a dichotomic search was used instead of CROSH to get an initial upper bound. The lower bound lb is set to the duration of the longest job/machine, whilst the upper bound ub in the dichotomic search is initialised by a greedy algorithm. We repeatedly solve the decision problem with a makespan fixed to $\frac{ub+lb}{2}$, updating lb and ub accordingly, until they have collapsed.

Figure 1 is a log-log plot of the average (a) time and (b) number of nodes taken by the two models to solve each of the 192 open shop instances. Next, in Table 1, we selected a subset of 11 of the hardest problems based on overall results, choosing problems for which at least one of the light and heavy models took over 20 seconds to solve. This subset consisted of one Gueret-Prins, 4 Taillard and 6 Brucker instances, respectively of order 10, 20 and 7-8. The results are presented in terms of average time and average nodes (where a model failed to prove optimality on a problem its time is taken as 3600s, so in these cases the average time is a lower bound).

Both Choco models proved optimality on all 11 instances. However, the light model is generally much faster (only slower on 1 of the 11 instances), even though it explores many more nodes. The Mistral light model was fastest on all problems (nodes for Mistral include those explored in dichotomic search, hence the difference). It is also of



(a) Time Comparison (b) Node Comparison

Fig. 1. Log-log plots of Choco Light vs Heavy models on open shop problems.

interest to note that on the hardest problems (6 Brucker instances), the heavy model was slightly quicker on average than the (Choco) light model at finding the optimal solution (310s vs 360s), but was 3 times slower at proving optimality once the solution had been found (385s vs 125s). This reinforces our belief that the weighted degree heuristic is adept at identifying the most contentious variables.

In order to better understand the importance of the heuristics to the two models we ran the same experiments but with the heuristics swapped, "Light-profile" and "Heavy-domwdeg". As can be seen in the table, swapping heuristics resulted in worse performance in both cases (failing to prove optimality on some problems for both), albeit more noticeably for combining profile with the light model.

Table 1. Results (Time) For Hard Open Shop Scheduling Problems

Instances	Light		Heavy		Light-profile		Heavy-domwdeg		Mistral	
	Time (s)	Nodes	Time (s)	Nodes	Time (s)	Nodes	Time (s)	Nodes	Time (s)	Nodes
GP10-01	6.1	4K	118.4	53K	2523.2	6131K	9.6	3K	**0.3**	3K
j7-per0-0	854.5	5.1M	1310.9	1.4M	979.1	4.3M	3326.7	2.6M	**327.0**	8.7M
j7-per10-2	57.6	0.4M	102.7	0.1M	89.5	0.4M	109.6	0.1M	**33.5**	1.2M
j8-per0-1	1397.1	6.4M	1973.8	1.5M	1729.3.	6.0M	3600.0	2.4M	**427.1**	10M
j8-per10-0	24.6	0.10M	30.9	0.02M	19.7	0.05M	68.0	0.06M	**17.6**	0.47M
j8-per10-1	275.2	1.3M	154.8	0.1M	92.8	0.3M	796.7	0.7M	**89.3**	2.3M
j8-per10-2	335.3	1.6M	651.4	0.6M	754.0	2.7M	697.1	0.6M	**93.1**	2.3M
tai-20-1	25.4	2K	27.5	3K	2524.7	1458K	34.4	3K	**3.9**	21K
tai-20-2	56.5	11K	178	42K	3600.0	2261K	81.9	10K	**14.1**	61K
tai-20-7	47.8	9K	60.9	11K	3600.0	2083K	63.7	8K	**9.5**	47K
tai-20-8	66.8	14K	108.3	25K	3600.0	2127K	84.8	11K	**8.2**	39K
Total Avg	286.1	1.3M	428.9	0.4M	1774.3	2.5M	806.6	0.6M	**93.0**	2.3M

Finally, we investigated whether the difference in restarting strategies might account for the improvement with the light model. We ran the Luby strategy on the light model and the geometric strategy on the heavy model. Again, in both cases, the change lead to a degradation in performance.

Search Heuristics We next assess the impact of the two aspects of the domwdeg heuristic, task size (dom) and weighted degree ($wdeg$), on the light model.

The following experiments were run using the Mistral light model with the same settings as before. However, due to the nature of wdeg and dom, all variable heuristics

were further randomized by randomly choosing from their top three choices when no tie existed. Furthermore the cutoff for dichotomic search was 30 seconds for each (lb ub) to allow all heuristics to achieve a good initial upper bound. We present our results in terms of average makespan found (mks), average percent proven optimal and the average time over the sample of 20 runs for each problem.

Table 2. Variable Heuristic Comparison: Open Shop Scheduling Problems

Instances	dom			wdeg			domwdeg		
	Mks	Opt(%)	Time(s)	Mks	Opt(%)	Time (s)	Mks	Opt(%)	Time (s)
Brucker	1019.7	89.2	1529.2	1019.5	100.0	381.4	1019.5	100.0	249.7
Taillard	1253.7	0.0	3600.0	1215.2	93.7	1423.9	1214.7	100.0	8.8
Total Avg	1113.4	53.0	2358.5	1097.7	97.0	798.4	1097.6	100.0	153.3

Table 2 presents our findings on the Brucker and Taillard instances of Table 1. The results clearly show the effectiveness of wdeg on these problems. It proved optimality on average 97% of the time, and in the cases where it failed to prove optimality the average makespan found was within one of the optimal value. However, as previously mentioned, it suffers from a lack of discrimination at the start of search. For example, in the tai-20-* problems there are 7600 variables all with an initial weighted degree of 1. Discrimination will only occur after the first failure which, given the size of the problems and the looseness of constraints, can occur deep in search, especially with a heuristic that is random up until at least one failure occurs.

The domain heuristic is relatively poor on these problems (only proving optimality 53% of the time and finding poor makespans for the tai-20-* problems). However, the addition of this information to the wdeg heuristic (*domwdeg*) results in 100% optimality. The domain factor has two benefits, it provides discrimination at the start of search, and it improves the quality of the initial weights learnt due to its fail firstness.

3.2 Job shop scheduling

We now compare the light model, implemented in Mistral, with a randomized restart algorithm used in [4], which is implemented in Ilog scheduler. It should be noted that we are not comparing with Beck's SGMPCS algorithm, but with the randomized restarts approach that was used as an experimental comparison by Beck. This algorithm is nearly identical to the heavy model introduced in Section 2.

All parameters for the algorithm are taken from [4], so the restart strategy is Luby with an initial failure limit of 1 (i.e there is no scale factor). The variable ordering heuristic is the profile heuristic described earlier. Randomization is added by randomly selecting with uniform probability from the top 10% most critical pairs of (machine, time point). Finally, the standard constraint propagation techniques for scheduling are used, such as time-table [19], edge-finding [18], and balance constraints [12]. However nogood recording from restarts isn't part of the algorithm. We ran experiments on the same cluster described earlier, using Ilog scheduler 6.2.

The Mistral parameters are the same as for the open shop problems, with the exception that we used a 300 second cutoff for the dichotomic search in order to achieve a

good initial upper bound. Furthermore, we used a static value ordering heuristic, where each pair of operations on a machine were ordered based on their relative position in their job (i.e. if precedences on two jobs state that t_i is second and t_j is fourth on their respective jobs, then b_{ij} will first branch on 0 during search, i.e. t_i precedes t_j). (We experimented with several value heuristics and this proved to be the best.)

Table 3 describes our results on 4 sets of 10 JSP instances proposed by Taillard [21]. The different sets have problems of different sizes (#jobs x #machines). The four sets are of size 20x15, 20x20, 30x15, 30x20, respectively. For each instance of each set, 10 randomized runs were performed, since problems were rarely solved to optimality within the cutoff. (This set of experiments took roughly 33 days of CPU time.)

We present our results in terms of averages over each set of problems. In particular, the average of the mean makespan found per problem in each set, the average of the best makespan found for each problem in each set, and average standard deviation.

Table 3. Job Shop Scheduling Problems

Instances	Scheduler			Mistral		
	Mean	Best	Std Dev	Mean	Best	Std Dev
tai11-20	1411.1	1409.9	11.12	**1407.3**	**1392.9**	24.35
tai21-30	**1666.0**	1659.0	13.51	1666.8	**1655.1**	22.91
tai31-40	1936.1	1927.1	18.67	**1921.6**	**1899.2**	37.79
tai41-50	2163.1	2153.1	17.85	**2143.3**	**2119.5**	42.71

As expected, since we used faster hardware, the results we obtained with Ilog Scheduler match, or improve slightly on the values reported in [4]. Both approaches perform similarly on the first two sets, although best solutions found by mistral are consistently better than Ilog scheduler. Mistral scales up better on the next two (larger) sets, indeed its average mean makespan for each set is better than Scheduler's average best makespan in both sets. It is interesting to note that the standard deviation is much larger for the lighter model. The down side is that it means our approach is less robust. However, it also means that using parallel computing should improve the lighter model more than it would improve Scheduler's results.

4 Conclusion

In this paper we have shown that, contrary to popular belief, disjunctive temporal problems (such as in the scheduling domain) can be efficiently solved by combining naive propagation with the generic weighted degree heuristic. Important additional factors in such an approach are restarting, nogood recording from restarts, a good method for finding an initial upper bound, and an element of randomization. We have shown that such an approach can often outperform the state of the art solvers for open shop and job shop scheduling problems.

However, our approach was not able to match the results of solution guided multi point constructive search (SGMPCS) for job shop scheduling problems [4]. It is our belief that a similar solution guided approach can be incorporated into our model to improve its performance, and this is the direction we intend to take our future work.

5 Acknowledgements

The authors would like to thank Hadrien Cambazard for his valuable comments, and Chris Beck for supplying the code for the job shop scheduling comparison. This work was supported by Science Foundation Ireland under Grant 05/IN/I886.

References

1. Malapert A, H. Cambazard, C. Guéret, N. Jussien, A. Langevin, and L-M. Rousseau. An Optimal Constraint Programming Approach to the Open-Shop problem. *submitted to INFORMS, Journal of Computing*, 2008.
2. P. Baptiste, C. Le Pape, and W Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming Techniques to Scheduling Problems*. Kluwer Academic Publishers, 2001.
3. J. C. Beck, A. J. Davenport, E. M. Sitarski, and M. S. Fox. Texture-Based Heuristics for Scheduling Revisited. In *AAAI/IAAI*, pages 241–248, 1997.
4. J. Christopher Beck. Solution-Guided Multi-Point Constructive Search for Job Shop Scheduling. *Journal of Artificial Intelligence Research*, 29:49–77, 2007.
5. F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais. Boosting Systematic Search by Weighting Constraints. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'04)*, pages 482–486, Valencia, Spain, August 2004.
6. Peter Brucker, Johann Hurink, Bernd Jurisch, and Birgit Wöstmann. A Branch & Bound Algorithm for the Open-shop Problem. In *GO-II Meeting: Proceedings of the Second International Colloquium on Graphs and Optimization*, pages 43–59, Amsterdam, The Netherlands, 1997. Elsevier Science Publishers B. V.
7. J. Carlier and E. Pinson. An Algorithm for Solving the Job-shop Problem. *Management Science*, 35(2):164–176, 1989.
8. The choco team. choco: an Open Source Java Constraint Programming Library. In *the Third International CSP Solver Competition*, pages 31–40, 2008.
9. M. S. Fox, N. M. Sadeh, and C. A. Baykan. Constrained heuristic search. In *IJCAI*, pages 309–315, 1989.
10. C. Guéret and C. Prins. A new Lower Bound for the Open Shop Problem. *Annals of Operations Research*, 92:165–183, 1999.
11. E. Hebrard. Mistral, a Constraint Satisfaction Library. In *the Third International CSP Solver Competition*, pages 31–40, 2008.
12. P. Laborie. Algorithms for Propagating Resource Constraints in AI Planning and Scheduling: Existing Approaches and new Results. *Artificial Intelligence*, 143(2):151–188, 2003.
13. C. Lecoutre, L. Sais, S. Tabary, and V. Vidal. Nogood Recording from Restarts. In *IJCAI*, pages 131–136, 2007.
14. M. Luby, A. Sinclair, and D. Zuckerman. Optimal Speedup of Las Vegas Algorithms. In *ISTCS*, pages 128–133, 1993.
15. I. Meiri. Combining Qualitative and Quantitative Constraints in Temporal Reasoning. In *AAAI*, pages 260–267, 1991.
16. E. Nowicki and C. Smutnicki. A Fast Taboo Search Algorithm for the Job Shop Problem. *Manage. Sci.*, 42(6):797–813, 1996.
17. E. Nowicki and C. Smutnicki. An Advanced Tabu Search Algorithm for the Job Shop Problem. *Journal of Scheduling*, 8(2):145–159, 2005.
18. W. Nuijten. *Time and Resource Constraint Scheduling: A Constraint Satisfaction Approach*. PhD thesis, Eindhoven University of Technology, 1994.

19. C. Le Pape. Implementation of Resource Constraints in ILOG SCHEDULE: A Library for the Development of Constraint-Based Scheduling Systems. *Intelligent Systems Engineering*, 3:55–66, 1994.
20. D. Y. Sha and Cheng-Yu Hsu. A new Particle Swarm Optimization for the Open Shop Scheduling Problem. *Computers & Operation Research*, 35(10):3243–3261, 2008.
21. E. Taillard. Benchmarks for Basic Scheduling Problems. *European Journal of Operations Research*, 64:278–285, 1993.
22. T. Walsh. Search in a Small World. In *IJCAI*, pages 1172–1177, 1999.
23. J-P. Watson and J. C. Beck. A Hybrid Constraint Programming / Local Search Approach to the Job-Shop Scheduling Problem. In *CPAIOR*, pages 263–277, 2008.