Integrating Mixed-Integer Optimisation and Satisfiability Modulo Theories: Application to Scheduling

M. Mistry and R. Misener*

Department of Computing, Imperial College London, South Kensington Campus, SW7 2AZ, UK

Abstract

One way to address multi-scale optimisation problems is by integrating logic and optimisation. For example, a scheduling problem may have two levels: (i) assigning orders to machines and (ii) sequencing orders on each machine. In a minumum cost model, assigning orders to machines is a mixed-integer optimisation problem, sequencing orders is a constraint satisfaction problem. The entire problem may be reformulated as either an optimisation or logic problem, but this misses the chance to use optimisation and logic synergistically. Hybrid optimisation/logic approaches have been developed combining mixed-integer linear programming (MILP) and constraint programming (CP), but CP requires specialised, bespoke constraints. We consider modifying the hybrid method by replacing CP with satisfiability modulo theories (SMT); SMT is a constraint satisfaction technique combining propositional satisfiability with a background theory. We find that a logic-based Benders decomposition approach combining MILP and SMT works very well on a minimum cost model for scheduling and performs significantly better than either MILP or SMT alone. But the hybrid MILP/SMT method is weaker than either MILP or SMT on a minimum makespan model.

Keywords

Mixed-integer programming, satisfiability modulo theories, scheduling, logic-based Benders decomposition

Introduction

A major challenge in process systems engineering is integrating: (i) long-term strategic planning decisions, (ii) medium-term tactical planning, and (iii) short-term scheduling (Maravelias and Sung, 2009). One way to address these multi-scale optimisation problems is by integrating logic and optimisation (Hooker and Ottoson, 2003). For example, a scheduling problem may have two levels: (i) assigning orders to machines and (ii) sequencing orders on each machine (Jain and Grossmann, 2001). In a minimum cost model, assigning orders to machines is a mixed-integer optimisation problem, sequencing orders is a constraint satisfaction problem. The entire problem may be reformulated as either an optimisation or logic problem, but this misses the chance to use optimisation and logic synergistically.

Hybrid optimisation/logic approaches have been developed combining mixed-integer linear programming (MILP) and constraint programming (CP), e.g. Jain and Grossmann (2001); Li and Womer (2008); Sitek (2014). The hybrid formulations usually use logic-based Benders decomposition (LBBD) (Hooker and Ottoson, 2003), a generalisation of Benders decomposition (Benders, 1962). The principles of Benders decomposition remain: we have a master problem and a subproblem which generates cuts if the solution from the master problem is infeasible. The difference is that LBBD requires a logic proof deriving an objective bound.

Hybrid MILP/CP methods are typically applied scheduling and its variants (Sitek, 2014). This is reasonable: CP is very good at assessing scheduling feasibility. The problem with hybrid MILP/CP is that, if the application does not have a suitable CP constraint, a hybrid method may be poor since bespoke CP constraints take full advantage of very specific mathematical structures. This manuscript is to test satisfiability modulo theories as an alternative to CP in the hybrid scheme.

Satisfiability Modulo Theories

Satisfiability modulo theories (SMT) is a constraint satisfaction technique combining propositional satisfia-

*r.misener@imperial.ac.uk; Tel: +44 (0) 20759 48315

bility (SAT) with a background theory (De Moura and Bjørner, 2008). A background theory is a set of axioms and symbols, e.g. the theory of arithmetic. An SMT solver consists of a SAT solver and a theory solver. The idea is to leverage the strength and robustness of modern SAT solvers to search for a feasible solution. The modelling framework exposed by SMT allows for Boolean variables to be used with background theory variables, e.g $z \to (x \geq 0)$ where x is continuous and z is Boolean, so SMT is a natural choice when logical decisions form a part of the system being modelled.

SMT assesses the satisfiability of a model and, if the model is satisfiable, the SMT solver returns a witness. If the model is unsatisfiable the SMT solver can return an unsatisfiable core, a mutually unsatisfiable subset of model constraints. An unsatisfiable core is a useful tool when addressing why a model does not behave how we expect or to understand why our model fails. Commonly used SMT solvers include Z3 (De Moura and Bjørner, 2008) and MathSAT (Cimatti et al., 2013).

Planning and Scheduling

See Table 1 for descriptions of the sets, parameters and variables mentioned in the formulations.

MILP Models

The MILP formulation follows the Hooker (2007) discrete time model; a continuous time formulation (Türkay and Grossmann, 1996), while smaller, is harder to solve for the models considered in this manuscript. The minimum cost model is:

$$\min \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}} F_{ij}x_{ijt} \tag{1}$$

$$\text{s.t.} \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}} x_{ijt} = 1, \qquad \forall j \in \mathcal{J} \tag{2}$$

$$\sum_{j\in\mathcal{J}}\sum_{t'\in\mathcal{T}_{ijt}} c_{ij}x_{ijt'} \leq C_i, \quad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{3}$$

$$x_{ijt} = 0, \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}, \quad t < r_j \text{ or } t > d_j - p_{ij} \tag{4}$$

$$x_{ijt} \in \{0,1\}, \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}, t \in \mathcal{T}. \tag{5}$$

Equation (2) requires that a task is only assigned to a single machine and only starts once. Equation (3) characterises the resource constraints using the set $\mathcal{T}_{ijt}$. Equation (4) limits the time windows for a given task as defined by r_j and d_j.

The makespan is the total schedule length. The minimum makespan model is similar to minimum cost, but minimum makespan requires an additional variable $M \geq 0$, a new constraint bounding the makespan M from below by the local makespan of each task:

$$M \geq \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}(t + p_{ij})x_{ijt}, \quad \forall j \in \mathcal{J}, \tag{6}$$

and a different objective:

$$\min\ M. \tag{7}$$

Table 1. Model symbols (Hooker, 2007).

Name	Description
Sets	
$\mathcal{I} = \{1,\ldots,m\}$	Facilities
$\mathcal{J} = \{1,\ldots,n\}$	Tasks
$\mathcal{T} = \{1,\ldots,p\}$	Discrete time points
$\mathcal{T}_{ijt} = \{t' \mid t - p_{ij} < t' \leq t\}$	Start times: j is in progress at t on i
$\mathcal{H} = \{1,\ldots,H-1\}$	Iteration indices; assume that we are at iteration H in the hybrid model
$\mathcal{J}_{hi} \subseteq \mathcal{J}$	Local assignment of tasks to i in iteration h in hybrid model
$\bar{\mathcal{J}}_{hi} \subseteq \mathcal{J}_{hi}$	Tasks mutually reponsible for infeasibility/local optimality in hybrid model
Parameters	
p_{ij}	Processing time of j on i
c_{ij}	Resource consumption of j on i
C_i	Resource capacity on i
r_j	Release time of j
d_j	Due time of j
F_{ij}	Cost of assigning j to i (min cost only)
Variables	
x_{ijt}	Assign j to i starting at t (MILP, binary)
x_{ij}	Assign j to i (hybrid, binary)
ζ_{ij}	Assign j to i (SMT, Boolean)
s_j	Start time of j (SMT & hybrid, continous)
c'_j	Resource 'postition' of j (SMT, continous)
M	Makespan (min makespan only, continous)

Logical Models

As in the MILP case, a discrete logical formulation can be very large when set $\mathcal{T}$ is large. But, for an SMT solver, a discrete formulation loses information and is therefore less favourable. Some inherent task properties are the release, due and processing times which are not directly present in the model. They would have to be inferred from the constraints given by Eqs. (3) and (4), and the set $\mathcal{T}_{ijt}$ and may weaken the SMT theory solver.

A continuous time minimum cost model is:

$$\min \sum_{j\in\mathcal{J}} f_j \tag{8}$$

$$\text{s.t. } s_j \geq r_j, \qquad \forall j \in \mathcal{J} \tag{9}$$

$$\zeta_{ij} \rightarrow (s_j \leq d_j - p_{ij}), \qquad \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{10}$$

$$\bigvee_{i\in\mathcal{I}} \left(\zeta_{ij} \wedge \bigwedge_{i'\neq i} \neg\zeta_{i'j} \right), \qquad \forall j \in \mathcal{J} \tag{11}$$

$$\bigwedge_{j\in\mathcal{J}'} \zeta_{ij} \rightarrow \left(\left(\sum_{j\in\mathcal{J}'} c_{ij} \leq C_i \right) \vee \bigvee_{\substack{j,j'\in\mathcal{J}' \\ j'\neq j}} (s_j + p_{ij} \leq s_{j'}) \right), \tag{12}$$

$$\forall i \in \mathcal{I}, \mathcal{J}' \in \mathbb{P}(\mathcal{J}) \setminus \emptyset$$

$$\zeta_{ij} \rightarrow (f_j = F_{ij}), \qquad \forall i \in \mathcal{I}, j \in \mathcal{J}. \tag{13}$$

The minimum makespan model is similar to the minimum cost model, but does not have variables f_j or parameters F_{ij}. Additional variable $M \geq 0$ represents the makespan and the minimum cost objective Eq. (8) is replaced with the makespan objective:

$$\min\ M. \tag{14}$$

We also introduce the constraint which implies that if task j is assigned to facility i then the makespan must be greater than or equal to the task's completion time:

$$\zeta_{ij} \rightarrow (M \geq s_j + p_{ij}), \quad \forall i \in \mathcal{I}, j \in \mathcal{J}. \tag{15}$$

This logical formulation is flawed because the total number of Eq. (12) constraints is exponential in the number of tasks and, even for small problems, model building time may become the bottleneck. Also, some of these constraints may provide redundant information, e.g. if a set of tasks cannot be scheduled on the same machine, any superset of these tasks cannot be scheduled.

We improve the logical formulation by interpreting time and resource consumption as separate dimensions and thereby convert the planning and scheduling problem to a generalised two dimensional bin packing problem (Garey et al., 1976). The difference between this conversion and 2BP is that the items (tasks) may not have the same height and width when they are placed in different bins (machines); the items are further constrained on one dimension (release and due times). We convert Eq. (12) by modelling task j as an item with height c_{ij} and width p_{ij} when placed on machine i. New variable $c'_j \geq 0$ with upper bounding constraint:

$$\zeta_{ij} \rightarrow \left(c'_j \leq C_i - c_{ij}\right), \quad \forall i \in \mathcal{I}, j \in \mathcal{J} \tag{16}$$

accounts for the new dimension and Eq. (12) is replaced with:

$$(\zeta_{ij} \wedge \zeta_{ij'}) \rightarrow \begin{cases} (s_j + p_{ij} \leq s_{j'}) \vee (s_{j'} + p_{ij'} \leq s_j) \\ \vee (c'_j + c_{ij} \leq c'_{j'}) \vee (c'_{j'} + c_{ij'} \leq c_j) \end{cases} \tag{17}$$

$$\forall i \in \mathcal{I}, j, j' \in \mathcal{J}, j < j'.$$

With this replacement, there are a quadratic rather than exponential number of constraints. This formulation is more meaningful when considering the SMT unsatisfiable core since the tasks are related on a pairwise basis, e.g. an unsatisfiable core consisting of pairs $\{(1,2),(1,3),(2,3)\}$ would mean that these three tasks are unsatisfiable when assigned to the same machine whereas the exponential can return any constraint that corresponds to a superset of $\{1,2,3\}$. The algorithm we implement for pure SMT bounds the objective seeking feasible solutions hence we do not get an unsatisfiable core, but for the hybrid models we can get unsatisfiable cores from which we derive cuts.

Novel Hybrid Model combining SMT and MILP

The hybrid MILP/SMT strategy splits the problem into: the master problem and the subproblem. The master problem optimises a less constrained problem; this optimisation solution is checked in the subproblem for correctness. If feasible then optimality is achieved, otherwise a cut is derived to reject the current incumbent (and possibly others) from the master problem and the process is repeated.

For planning and scheduling, we adapt the Hooker (2007) LBBD method developed for hybrid MILP/CP. The formulations, cuts and relaxations listed below were all formulated by Hooker (2007), we describe the differences in adapting the approach to hybrid MILP/SMT. The problems are solved with a master problem (MILP) that assigns tasks to facilities and a subproblem (SMT) that assesses the feasibility (for minimum cost) or local optimality (for minimum makespan). The solution process iterates by solving the master problem for an assignment and then, having fixed these assignments, solving the subproblem and repeating until the termination criteria is satisfied. On each iteration, the subproblem derives Benders cuts to be added to the master problem, these cuts prevent the current assignment from being reassessed and are typically strong enough to prune large amounts of the search space. Hooker (2007) states that 'experience shows that it is important to include a relaxation of the subproblem'; we use the same relaxations

when assessing the hybrid MILP/SMT approach.

Only the master problems are formulated here, the subproblems consist of constraints associated with the start times and resource limits where the Boolean variables are fixed according to the master problem assignment. For minimising cost, the master problem is:

$$\min \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} F_{ij}x_{ij} \tag{18}$$

$$\text{s.t.} \sum_{i\in\mathcal{I}} x_{ij} = 1, \qquad \forall j \in \mathcal{J} \tag{19}$$

$$\sum_{j\in\bar{\mathcal{J}}_{hi}}(1-x_{ij}) \geq 1, \qquad \forall i \in \mathcal{I}_h, h \in \mathcal{H}, \tag{20}$$

$$\sum_{j\in J(t_1,t_2)} p_{ij}c_{ij}x_{ij} \leq C_i(t_2-t_1), \tag{21}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}. \tag{22}$$

The binary variables x_{ij} represent task j being assigned to machine i. The master problem only contains the assignment variables and costs; task start times are in the subproblem. The x_{ij} values in a master problem solution have a one-to-one correspondence with subproblem variables ζ_{ij}, i.e. $x_{ij} = 1 \iff \zeta_{ij} = \text{True}$. Equation (20) are the Benders cuts added on each iteration. Note that the task set is $\bar{\mathcal{J}}_{hi}$ which is derived by greedily filtering tasks from $\mathcal{J}_{hi}$ while keeping the local assignment infeasible. The SMT unsatisfiable core may not necessarily return all unsatisfiable facilities under the local assignment (it is more likely that the core will only be associated with a single machine), be the core is mutually unsatisfiable therefore it will satisfy the properties we want in $\bar{\mathcal{J}}_{hi}$. Equation (21) is the subproblem relaxation where $J(t_1,t_2)$ contains all tasks that are released at or after t_1 and are due at or before t_2, there are a finite number of (t_1,t_2) pairs which can be derived from the combinations of release and due times across all tasks. Hooker (2007) gives an intuition for this relaxation by referring to $p_{ij}c_{ij}$ as the 'energy' of task j on facility i and $C_i(t_2-t_1)$ as the total energy available in time window $[t_1,t_2]$ on facility i. Therefore the total energy of the tasks $j' \in J(t_1,t_2)$, $\sum_{j'} p_{ij'}c_{ij'}$, must not exceed the total available energy. Given the bin packing formulation, an alternative interpretation is that $p_{ij}c_{ij}$ is the area of task (item) j in facility (bin) i and $C_i(t_1-t_2)$ is the total available area of that window.

The master problem for minimising makespan differs on the objective, Benders cuts and subproblem relaxation. As for the MILP and logical models, we introduce the variable $M \geq 0$ and objective:

$$\min\ M. \tag{23}$$

The relaxation added to the problem is:

$$C_iM \geq \sum_{j\in\mathcal{J}} c_{ij}p_{ij}x_{ij}, \quad \forall i \in \mathcal{I}. \tag{24}$$

The relaxation is similar to the minimum cost relaxation however here the 'total area' is defined by the makespan. If the assignment is feasible, the Benders cuts take the form:

$$M \geq M^*_{hi} - \sum_{j\in\bar{\mathcal{J}}_{hi}}(1-x_{ij})p_{ij}, \forall i \in \mathcal{I}_h, h \in \mathcal{H}, \tag{25}$$

where M^*_{hi} is the optimal makespan achieved on facility i with the assignment in iteration h. Similarly to minimum cost, the set $\bar{\mathcal{J}}_{hi}$ represents a subset of all tasks assigned to i however here it is a set that results in M^*_{hi} and removal of a task would cause M^*_{hi} to change. If the assignment is infeasible, we add an Eq. (20) cut to reject it. In SMT applying an algorithm to find such a set is unnecessary since the unsatisfiable core results in one. If the deadlines of the tasks in the set $\bar{\mathcal{J}}_{hi}$ are not all the same, we can add the stronger Benders cut (note w_{hi} is new):

$$M \geq M^*_{hi} - \sum_{j\in J_{hi}}(1-x_{ij})p_{ij} - w_{hi}, \tag{26}$$

$$w_{hi} \leq \left(\max_{j\in J_{hi}}\{d_j\} - \min_{j\in J_{hi}}\{d_j\}\right)\sum_{j\in J_{hi}}(1-x_{ij}), \tag{27}$$

$$w_{hi} \leq \max_{j\in J_{hi}}\{d_j\} - \min_{j\in J_{hi}}\{d_j\}, \tag{28}$$

$$w_{hi} \geq 0, \tag{29}$$

for all $i \in \mathcal{I}, h \in \{1,\ldots,H-1\}$.

Numerical Results

To solve the MILP and SMT models we used Gurobi 6.0.3 and Z3 (De Moura and Bjørner, 2008), respectively. The MILP models are in Pyomo (Hart et al., 2011, 2012); all further implementations are in Python (Z3 has a Python API). SMT assesses satisfiability rather than optimality, so we model the objective with a constraint that is iteratively tightened via the last objective found. All test cases were run on a HP EliteDesk 800 G1 TWR with 16GB RAM and an Intel® Core™ i7-4770 @ 3.40Ghz running Ubuntu 16.04.1 LTS. The test set, originally generated by Hooker (2007), can be found online[1]. There are 335 total instances from 4 classes: 195 from 'c', 50 from 'de', 40 from 'df' and 50 from 'e'. The subsequent analysis consists of performance profiles

[1] `http://web.tepper.cmu.edu/jnh/instances.htm`

(Dolan and Moré, 2002) and average runtimes. We say that an instance *terminates* if it proves infeasibility or converges within the timelimit to the optimal solution. The performance profiles discard instances with fewer than 18 tasks as toy problems. Results for these toy test cases may be biased towards implementation opposed to algorithm performance. The total number of problems after discarding toy instances is 230 among which there are 135 'c', 30 'de', 30 'df' and 35 'e' instances. The average run times are given for a subset of the 'c' instances these instances were chosen as they cover the boundary at which some solvers begin to time out and are indicative of how much of an improvement can be achieved by using an alternative. Hooker (2007) found that a hybrid MILP/CP method outperforms both MILP and CP independently with respect to both in speed and in tractability of problems. We see if a similar result can be found with the use of hybrid MILP/SMT.

We analyse the results of minimising costs first. Recall that Figure 1 discards problems with fewer than 18 tasks as toy instances. The hybrid implementation is fastest for more than half the test set and can solve close to 90% of the problems (nearly all tractable problems). We see that MILP is able to solve more problems in a shorter time than SMT however, in terms of tractability, the number of problems that are tractable by both approaches are fairly similar (about 60%).

The average running time for some of the 'c' instances with 2, 3 and 4 facilities is presented in Table 2, we exclude the larger instances as there are a fair amount of timeouts beyond the instances shown. The hybrid method is clearly superior for larger instances. SMT appears to be slightly better than MILP, but the hybrid formulation with the relaxation outperforms both.

The hybrid algorithm performs well for this objective because in each iteration we seek the next best assignment. This is handled by the MILP solver which is able to efficiently find it since it is not constrained by resource consumption or release and due times. The subproblem just has to check a series of independent resource constrained scheduling problems. Therefore we delegate parts of the problem in the correct locations.

We now analyse the makespan minimisation results. The Figure 2 performance profile, similarly to the minimum cost tests, discards instances with fewer than 18 tasks as toy problems. SMT and MILP are comparable in terms of the number of problems they can solve fastest, but SMT solves more problems in the hour. The hybrid method performs less well than MILP or SMT. Table 3 records the average running time for some of the 'c' instances with 2, 3 and 4 facilities. Here, the hybrid approach performs worse than SMT and, for some of the larger instances, performs at least three times worse than MILP. SMT seems to scale quite well as for the larger instances, of Table 3, SMT is at least seven times faster than either of the other two methods.

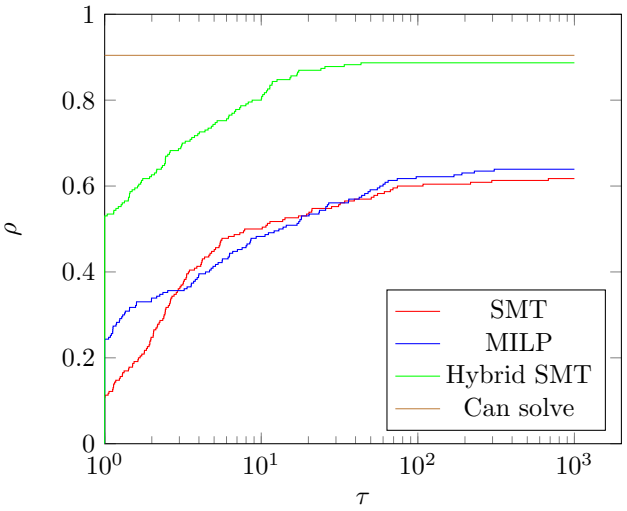

Figure 1. Minimum cost performance profile for all Hooker (2007) problems with at least 18 tasks (230/335 in the test set: 135 'c', 30 'de', 30 'df', 35 'e').

Table 2. Average minimum cost running times for the 5 'c' instances with $\#\mathcal{I}$ facilities and $\#\mathcal{J}$ tasks. A '+' indicates that at least one instance timed out (limit 3600s).

$\#\mathcal{I}$	$\#\mathcal{J}$	MILP	SMT	Hybrid SMT
2	14	0.78	0.33	0.71
	16	5.55	2.61	4.09
	18	82.38	8.57	9.02
	20	167.50	28.93	2.20
	22	1160.11+	1070.26+	789.67+
3	16	5.38	1.48	7.84
	18	65.90	3.68	12.06
	20	739.59+	11.53	4.09
	22	945.38+	31.37	8.07
	24	1886.62+	695.86	24.60
4	18	3.90	3.22	6.52
	20	36.37	13.14	5.56
	22	39.26	69.95	8.99
	24	1800.25+	1523.49+	75.29
	26	2964.38+	2805.87+	35.09

The SMT algorithm performs well here because there is a tight coupling between constraints and objective, i.e. past knowlegde from the last iteration could be used in the current iteration. The hybrid algorithm does not

perform as well because the MILP solver only learns about the feasible space through the Benders cuts hence the idea of a next best assignment may not have as much of an effect in the early iterations. Also the independent analysis of each assignment prevents the SMT solver from learning about relations across facilities.

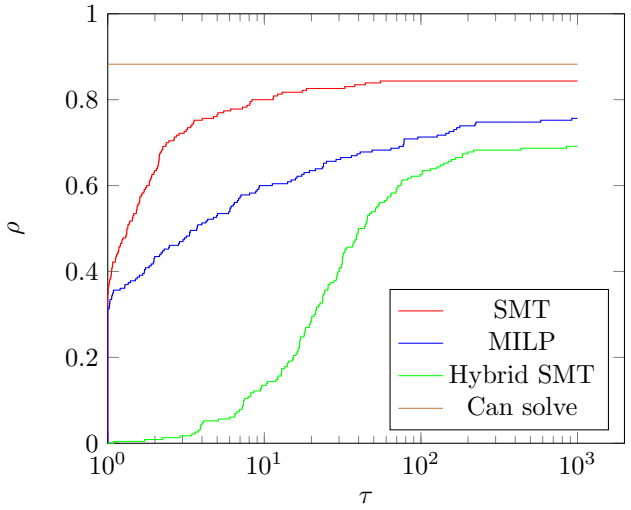

Figure 2. Minimum makespan performance profile for all Hooker (2007) problems with at least 18 tasks (230/335 in the test set: 135 'c', 30 'de', 30 'df, 35 'e').

Table 3. Average minimum makespan running times for 5 'c' instances with $\#\mathcal{I}$ facilities & $\#\mathcal{J}$ tasks. A '+' indicates that at least one instance timed out (limit 3600s).

$\#\mathcal{I}$	$\#\mathcal{J}$	MILP	SMT	Hybrid SMT
2	14	0.32	0.18	4.37
	16	9.47	3.05	82.40
	18	130.90	2.69	289.86
	20	769.88+	2.58	309.85
	22	1874.07+	102.07	1713.52+
3	16	0.65	0.54	9.65
	18	137.98	1.71	38.68
	20	9.08	0.81	13.68
	22	402.45	50.47	1540.65+
	24	499.24	70.05	1457.06+
4	18	0.82	0.66	14.22
	20	1.07	0.55	9.92
	22	12.71	3.59	34.98
	24	59.97	8.36	652.09
	26	1447.05+	7.66	275.57

Conclusions

This manuscript considers a logic-based Benders decomposition technique combining MILP and SMT; prior LBBD approaches use constraint programming rather than SMT. We find that hybrid MILP/SMT techniques are significantly stronger than either technique individually on minimum cost models of scheduling, but that the hybrid is weaker for minimum makespan.

Acknowledgments

The support of the EPSRC Centre for Doctoral Training in High Performance Embedded and Distributed Systems (HiPEDS, EP/L016796/1) and a Royal Academy of Engineering Research Fellowship to R.M. is gratefully acknowledged.

References

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252.

Cimatti, A., Griggio, A., Schaafsma, B., and Sebastiani, R. (2013). The MathSAT5 SMT Solver. In *TACAS*, volume 7795 of *LNCS*. Springer.

De Moura, L. and Bjørner, N. (2008). *Z3: An Efficient SMT Solver*, pages 337–340. Springer.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Math Program*, 91(2):201–213.

Garey, M. R., Graham, R. L., Johnson, D. S., and Yao, A. C.-C. (1976). Resource constrained scheduling as generalized bin packing. *J Combin Theory, Ser A*, 21(3):257 – 298.

Hart, W. E., Laird, C., Watson, J.-P., and Woodruff, D. L. (2012). *Pyomo–optimization modeling in Python*, volume 67. Springer Science & Business Media.

Hart, W. E., Watson, J.-P., and Woodruff, D. L. (2011). Pyomo: modeling and solving mathematical programs in Python. *Math Progam Comput*, 3(3):219–260.

Hooker, J. N. (2007). Planning and scheduling by logic-based Benders decomposition. *Oper Res*, 55(3):588–602.

Hooker, J. N. and Ottoson, G. (2003). Logic-based Benders decomposition. *Math Progam*, 96(1):33–60.

Jain, V. and Grossmann, I. E. (2001). Algorithms for Hybrid MILP/CP Models for a Class of Optimization Problems. *INFORMS J Comput*, 13(4):258–276.

Li, H. and Womer, K. (2008). Scheduling projects with multi-skilled personnel by a hybrid MILP/CP Benders decomposition algorithm. *J Sched*, 12(3):281–298.

Maravelias, C. T. and Sung, C. (2009). Integration of production planning and scheduling: Overview, challenges and opportunities. *Comput Chem Eng*, 33(12):1919 – 1930.

Sitek, P. (2014). A hybrid CP/MP approach to supply chain modelling, optimization and analysis. In *Computer Science and Information Systems (FedCSIS)*, pages 1345–1352.

Türkay, M. and Grossmann, I. E. (1996). Logic-based MINLP algorithms for the optimal synthesis of process networks. *Comput Chem Eng*, 20(8):959 – 978.