Logic-based Benders Decomposition for Alternative Resource Scheduling with Sequence Dependent Setups

Tony T. Tran and **J. Christopher Beck**[1]

Abstract. We study an unrelated parallel machines scheduling problem with sequence and machine dependent setup times. A logic-based Benders decomposition approach is proposed to minimize the makespan. This approach is a hybrid model that makes use of a mixed integer programming master problem and a specialized solver for travelling salesman subproblems. The master problem is used to assign jobs to machines while the subproblems obtain optimal schedules on each machine given the master problem assignments. Computational results show that the Benders model is able to find optimal solutions up to six orders of magnitude faster as well as solving problems six times the size previously possible with a mixed integer programming model in the literature and twice the size that a branch-and-bound algorithm can solve for similar problems. We further relax the Benders decomposition to accept suboptimal schedules and demonstrate the ability to parameterize solution quality while outperforming state-of-the-art metaheuristics both in terms of solution quality and mean run-time.

1 Introduction

Many practical scheduling problems exhibit alternative machines and sequence dependent setup times: a job may be assigned to one of a set of machines and a minimum time must elapse between consecutive jobs executing on the same machine. For example, reactors in a chemical plant must be cleaned when changing from processing one mixture to another. The cleaning times may depend on which job comes before the cleaning and which comes after. If the preceding chemical affects the succeeding one, longer cleaning may be required to ensure that the reactor is properly prepared. The products processed in the reverse order may require a shorter cleaning because the offending product in the previous example may not suffer the same contamination risks. Further examples can be found in the plastic, glass, paper and textile industries where setup times of significant length exist [2, 11].

We study the unrelated parallel machines scheduling problem (PMSP) with machine and sequence dependent setup times. In the PMSP, jobs must be assigned to one of a set of alternative resources and jobs assigned to the same resource have setup times defined as the time that must elapse between the end of one job and the start of the next. The setup time is sequence and machine dependent in that the elapsed time between jobs will differ depending on the sequence and machine to which the pair of jobs is assigned. We concern ourselves with minimizing the makespan or the maximum completion time of a schedule as the objective function. Using the three-field notation[13], this problem can be denoted as (R|sds|C$_{max}$).

We develop an exact method to solve the PMSP using a logic-based Benders decomposition approach originally developed to verify logic circuits [16]. The Benders decomposition makes use of a mixed integer program (MIP) master problem and a specialized traveling salesman problem (TSP) solver for the subproblems. The MIP model is used to find machine assignments and lower bounds on the makespan of each machine, while the sub-solvers sequence the jobs on machines. The Benders decomposition is able to find and prove optimal schedules up to six orders of magnitude faster than a MIP model in the literature and twice the size that a branch-and-bound algorithm can solve for similar problems. We also introduce a relaxation on optimality in the Benders decomposition to allow the model to find feasible schedules for larger problems. Although the model is no longer an exact method, we are able to provide guarantees on solution quality and out-perform existing metaheuristic approaches.

2 Background

In this section, we define the PMSP with sequence and machine dependent setup times. A review of related work is presented and an existing MIP model to solve the PMSP is defined.

2.1 Problem Definition

In the PMSP, a set of jobs, N, are to be scheduled on a set of machines, M, with the objective of minimizing the makespan. Each job has a processing time p_{ij}, the time to process job j on machine i. The machines in this system are unrelated, i.e., a job j can have a processing time greater than job k on one machine, but the reverse could be true on another machine. There is a sequence and machine dependent setup time, s_{ijk}, which is the time that must elapse before a machine can begin processing job k if job j precedes it on machine i. The setup times are assumed to follow the triangle inequality $s_{ijk} \leq s_{ijl} + s_{ilk}$. The goal of the problem is to determine how to assign jobs to machines and then sequence these jobs on each machine to minimize the makespan.

2.1.1 Mixed Integer Programming Model

A MIP model used to find optimal solutions for the unrelated PMSP with setup times is presented by various researchers [15, 21]. This formulation is based on the model proposed by Guinet [14] with different objectives of total completion time or total tardiness.

[1] University of Toronto, Canada, email: {tran,jcb}@mie.utoronto.ca

$$\begin{aligned}
\min \quad & C_{max} & & \\
\text{s.t.} \quad & \sum_{j=0, j\neq k}^{N} \sum_{i=1}^{M} x_{ijk} = 1 & k \in N & \quad (1) \\
& \sum_{j=0, j\neq h}^{N} x_{ijh} = \sum_{k=0, k\neq h}^{N} x_{ihk} & h \in N;\, i \in M & \quad (2) \\
& C_k \geq C_j + \sum_{i=1}^{M} x_{ijk}(s_{ijk} + p_{ik}) + V(\sum_{i=1}^{M} x_{ijk} - 1) & & \\
& & j \in N;\, k \in N & \quad (3) \\
& \sum_{j=0}^{N} x_{i0j} = 1 & i \in M & \quad (4) \\
& C_j \leq C_{max} & j \in N & \quad (5) \\
& C_0 = 0 & & \quad (6) \\
& C_j \geq 0 & j \in N & \quad (7) \\
& x_{ijk} \in \{0, 1\} & j, k \in N; i \in M & \quad (8)
\end{aligned}$$

where

C_{max}:	Maximum completion time (makespan)
C_j:	Completion time of job j
x_{ijk}:	1 if job k is processed directly after job j on machine i
V:	A large positive number

Constraint (1) ensures that each job is scheduled on a single machine only and after exactly one other job. Constraint (2) ensures that each job cannot be preceded or succeeded by more than one job. Constraint (3) sets the completion times of each job such that if job j precedes job k, job k cannot also precede job j. If job k is processed directly after job j, $\sum_{i=1}^{M} x_{ijk} - 1 = 0$ and the constraint makes it so that $C_k \geq C_j + s_{ijk} + p_{ik}$ with i being the machine on which the two jobs are assigned. If job k is not scheduled directly after job j on a machine, $\sum_{i=1}^{M} x_{ijk} - 1 = -1$ and the large V term makes the constraint redundant. Constraint (4) guarantees that only one job can be scheduled first on each machine. Constraint (5) sets the makespan to be at least as large as the largest completion time of all jobs. Constraint (6) sets the completion time of job 0, an auxiliary job used to enforce the start of a schedule, to zero.

2.2 Related Work

The PMSP with sequence and machine dependent setup times is strongly NP-hard because the single machine scheduling problem with sequence dependent setup times (1|sds|C*max*) is equivalent to a traveling salesman problem (TSP) [5]. Thus, the PMSP with setup times can be thought of as an allocation and routing problem where cities are allocated to salesmen who then must find their own tour. Even in the case with identical machines (P|sds|C*max*), the problem is strongly NP-hard [11]. The combinatorial complexity of the PMSP has resulted in little research in exact optimization methods.

The MIP model in Section 2.1.1 was used to benchmark the quality of heuristics for small problem instances. Previous researchers solved problems of 8 jobs and 4 machines and 9 jobs and 2 machines [15, 21]. They found that larger problems were intractable and the MIP model could not be used to find optimal schedules. For example, Helal et al. [15] found the problem with 9 jobs and 2 machines on average required 5 hours to find optimal solutions.

Most research has been focused on the PMSP with identical machines [9, 18]. Research on the PMSP with unrelated machines has concentrated on the problem without setup times. An exact branch-and-bound algorithm was developed by Lancia [19] to minimize makespan for a problem with two parallel machines. A genetic algorithm, simulated annealing, and tabu search were compared by Glass et al. [12]. They conclude that tabu search is capable of finding the best solutions in short periods of time (20 seconds), but the performance comparison for a larger time limit (100 seconds) shows no clear distinction between the three local search methods.

Rocha et al. [22] look at exact algorithms for the PMSP with sequence dependent setup times. In their study, jobs have due dates and the objective is to minimize a function of makespan and total tardiness. They compare their branch-and-bound algorithm against two MIP models and show that their model is capable of optimally solving problems of size up to 30 jobs. However, as the problem sizes increase, heuristic methods outperform branch-and-bound.

Al-Salem [1] developed the *partitioning* heuristic to solve large instances of the PMSP with setup times. The partitioning heuristic makes use of a constructive and improvement heuristic to assign jobs to machines and a TSP-like heuristic to sequence them. Rabadi et al. [21] presented a metaheuristic called Meta-RaPS. The heuristic is a modified COMSOAL (Computer Method of Sequencing Operations for Assembly Lines) [3] approach with the goal of minimizing makespan. Direct comparisons were made between Meta-RaPS and the partitioning heuristic on problems up to 120 jobs and 12 machines where Meta-RaPS was found to be 10% better in some cases. Helal et al. [15] developed a tabu search to solve the same problem and also compared their model to the partitioning heuristic. The tabu search was found to be up to 8% better than the partitioning heuristic on the same sized instances. An ant colony optimization (ACO) method was implemented and shown to perform better than the partitioning heuristic, tabu search, and Meta-RaPS [4]. In this study, all four heuristics were tested on instances up to size of 120 jobs and 8 machines where the performance was based on the difference between the makespan found and a simple lower bound. ACO was found to produce schedules with makespans within 1% to 7% deviation of the lower bound with Meta-RaPS close behind.

Focacci et al. [10] proposed a two-phase algorithm based on constraint programming (CP) to optimize a combination of the makespan and the sum of setup times for a similar problem to the PMSP with sequence dependent setup times. Jobs consist of multiple activities and precedence constraints exist between these activities. In the first phase of the algorithm, a time-limited, incomplete branch-and-bound method is used to find solutions with small makespan. The second phase minimizes the sum of setup times with the constraint that any schedule found must have a makespan equal to or less than the makespan found in the first phase. They run their model on problems of up to 16 jobs, each consisting of 12 activities, and 16 machines.

A similar class of problems to the PMSP are the min-max vehicle routing problems (VRP). In the VRP, multiple vehicles must travel between many nodes to service all customers. Generally, this class of problems corresponds to the PMSP with identical machines and 0 processing time durations. Valle et al. [23] present a branch-and-cut algorithm for the VRP, but show that even with only two vehicles, the algorithm is unable to solve most instances they tested. They improve their results by implementing a column generation heuristic but report duality gaps of 10% or more on many of the test instances. For the class of heterogenous vehicles, Baldacci et al. [6] provide a survey. They show various different VRP problems and solution approaches. Benders decomposition has been applied to these class of problems before[7], but to the authors' knowledge, the use of logic-based Benders decomposition has not.

3 Logic-Based Benders Decomposition

Hooker first used a partitioning algorithm called logic-based Benders decomposition to verify logic circuits [16]. The partitioning algorithm was later developed to look at the PMSP problem without setup times where the objective was to either minimize cost, makespan or total tardiness [17].

We make use of the logic-based Benders decomposition framework to solve the PMSP problem with setup times. The problem can be decomposed into an assignment master problem and sequencing subproblems. In the master problem, the jobs are assigned to machines. This assignment results in multiple subproblems, one per machine, where each subproblem requires the sequencing of the assigned jobs. A MIP model is presented for the master problem. Sequencing is accomplished in the subproblem with a TSP solver.

3.1 Assignment Master Problem

The MIP formulation of the master problem is a relaxation of the PMSP with setup times. In this relaxation, jobs are assigned to machines, but instead of requiring a single sequence of jobs on each machine, many small sequences are allowed which partition the jobs assigned to the machine. The setup times are calculated for each subsequence; their sum is a lower bound on the total setup time on a machine. This assignment and subsequencing leads to an infeasible schedule if multiple subsequences are created. However, the relaxation gives a tighter lower bound than if setup times are ignored, while being significantly easier to solve than the full problem. The master problem is as follows

$$
\begin{array}{lllr}
\min & C_{max} & & \\
\text{s.t.} & \sum_{j \in N} x_{ij} p_{ij} + \xi_i \leq C_{max}, & i \in M & (9) \\
& \sum_{i \in M} x_{ij} = 1, & j \in N & (10) \\
& \xi_i = \sum_{j \in N} \sum_{k \in N, k \neq j} y_{ijk} s_{ijk}, & i \in M & (11) \\
& x_{ik} = \sum_{j \in N} y_{ijk}, & k \in N;\, i \in M & (12) \\
& x_{ij} = \sum_{k \in n} y_{ijk}, & j \in N;\, i \in M & (13) \\
& \mathit{cuts} & & (14) \\
& x_{ij} \in \{0; 1\}, & j \in N;\, i \in M & (15) \\
& 0 \leq y_{ijk} \leq 1, & j, k \in N;\, i \in M & (16)
\end{array}
$$

where

C_{max}:	Makespan of the master problem
ξ_i:	Total setup time incurred from all sequences on machine i
x_{ij}:	1 if job j is processed on machine i
y_{ijk}:	1 if job k is processed directly after job j on machine i

The makespan on each machine with the relaxed setup times is defined in constraint (9) as the summation of processing times for all jobs that are assigned to that machine and the relaxed total setup time. Constraint (10) ensures that each job is assigned to exactly one machine. Constraint (11) assigns the relaxed setup time of a machine i, ξ_i, to be a lower bound on the additional time required from the sequencing of jobs, y_{ijk}, and their respective setup times, s_{ijk}. The relaxation of setup times allows, instead of a sequence of jobs from the first to last job processed on a machine, smaller sequences independent of each other. For example, given jobs j_1, j_2, j_3, j_4, and j_5, a feasible sequence is [*start* - j_1 - j_2 - j_3 - j_4 - j_5 - *end*]. However, (12) and (13) set each job to have exactly one other job scheduled directly before and after it without restricting sub-cycles, as was seen in constraint (3). It is, therefore, possible to assign two sequences, such as [*start* - j_1 - j_2 - j_3 - *end*] and [j_4 - j_5 - j_4 - j_5...]. Constraint (14) are *cuts* added to the master problem from the subproblem each time an infeasible solution is found. In the first iteration of the master problem, the set of *cuts* is empty.

The master problem formulation is equivalent to solving the (R|sds|C*max*), but instead of solving for the exact single sequence of jobs to process on a machine, many subsequences are allowed. The relaxation creates a tight lower bound for the actual makespan of a machine and is similar to solving the assignment problem. The makespan found from solving the master problem may be infeasible given a proper sequencing of jobs.

3.2 Sequencing Subproblem

Once a solution of the master problem is found, the set of jobs to schedule on each machine is known. These sets of jobs create $|M|$ subproblems, one for each machine. In this section, the TSP subproblem formulation is presented. The subproblem will create a sequence of jobs on a machine such that the makespan is minimized.

We know that the sequencing of jobs on a single machine is equivalent to a TSP with directed edges, also known as an asymmetric TSP. In the TSP representation, jobs are the nodes and distances between nodes are the setup time between the two connected jobs plus the processing time of the preceding job. Therefore, traveling along an edge from node a to node b will contribute the processing time of job a and the setup time from job a to job b. By representing the problem in such a way, the setup time problem on a single machine is equivalent to a TSP and a cycle of a TSP from a start node to all other nodes and back to the initial start node is the sequence of jobs with the distance traveled being the makespan.

We also experimented with a CP model in the subproblem and found it better to use a TSP solver which has been optimized to solve such problems in place of a generic CP solver which is more expressive, but not as good at solving this problem.

3.2.1 Feasible Schedules from the Subproblem

Solving the sequencing subproblem is guaranteed to produce a feasible schedule because we do not impose the constraint on the makespan from the master problem. Often with Benders decomposition, a model searches in the infeasible region of solutions and the first feasible solution found is the optimal one. In our logic-based Benders decomposition approach, while the search is performed in the infeasible region, the sequencing subproblem solves the local schedule once the jobs are allocated to machines. The solution from the subproblem creates a feasible schedule with a makespan equal to the largest makespan found across all subproblems. Therefore, it is possible to stop the solving at any time after the first complete iteration and obtain a feasible schedule. This schedule may not be optimal if the problem solving is stopped prematurely. However, it is possible to compare this value against the makespan found from the most recent master problem solution to calculate an upper bound on how far the current schedule is from optimality. Therefore, the Benders decomposition will store the best solution found so far. At the completion of all subproblems during an iteration, the schedule created will be compared to the best solution found and updated if necessary.

3.3 Cuts

If the makespan found in the subproblem is less than or equal to the master problem's makespan, C_{max}, then this subproblem is globally feasible and no cuts are added to the master problem. In the case where the makespan found is greater than C_{max}, a cut is created and added to the master problem. The master problem is then re-solved with the cuts from each sub-problem.

To define the cut, we introduce the value $maxPre_{hj}$, the maximum setup time if job j directly succeeds another job that is assigned to the same machine in iteration h. That is,

$$maxPre_{hj} = \max_{k \in N_i^h; k \neq j} (s_{ikj}),$$

where N_i^h is the set of jobs assigned to machine i in iteration h. We define θ_{hij} to be an upper bound on the effect of a job j to a schedule when assigned to machine i on iteration h and set it as

$$\theta_{hij} = p_{ij} + maxPre_{hj}.$$

The proposed cut is then

$$C_{max} \geq C_{max}^{hi*} - \sum_{j \in N_i^h} (1 - x_{ij})\theta_{hij}.$$

Here, C_{max} is the makespan variable in the master problem and C_{max}^{hi*} is the makespan found in iteration h when solving the subproblem for machine i. The cut states that the future solutions of the master problem can only decrease the makespan if another assignment of jobs is given. If the same assignment is given to the subproblem, the x_{ij} variables that are part of this cut will all equal to 1. If this is the case, then $(1 - x_{ij}) = 0$ for all j and the makespan of the subproblem becomes a lower bound on C_{max}. When a different assignment is made and at least one of the x_{ij} variables that previously had a value of 1 is 0, the cut enforces that the makespan in a future iteration must be bounded by the makespan found in the subproblem reduced by the corresponding θ_{hij} value(s).

A valid cut must adhere to two conditions: the cut removes the current solution from the master problem and does not eliminate any globally optimal solutions [8]. The cut does remove the current solution from the master problem since using the same assignment requires an increase in the makespan variable. To show that the cut does not remove a global optimal solution, we prove that θ_{hij} is a true upper bound on the contribution of a job to an optimal schedule.

Theorem 1 *The proposed cut is guaranteed to provide a lower bound on the possible makespans in future iterations.*

Proof We show that the cut is a lower bound by assuming that a schedule violating the cut exists and then showing a contradiction.

Given a set of jobs, N, assigned to machine i in the current iteration, let C_N be the optimal makespan on machine i. We define two disjoint subsets $\bar{N} \cup \hat{N} = N$. Assume that in a subsequent iteration the jobs in $\bar{N}$ are assigned to machine i, their minimal makespan is $C_{\bar{N}}$, and contrary to our theorem:

$$C_{\bar{N}} < C_N - \sum_{j \in \hat{N}} \theta_{hij} \quad (17).$$

Given the schedule corresponding to $C_{\bar{N}}$, it is possible to construct a schedule containing all of N jobs by placing each job in $\hat{N}$, one by one, at the end of the partial schedule. Using Pre_j as the setup time required to schedule job j at the end of the current schedule, we know the makespan of the constructed schedule to be

$$C_{\bar{N}} + \sum_{j \in \hat{N}} (p_{ij} + Pre_j).$$

This schedule of all N jobs must have a makespan greater than or equal to C_N. However, $Pre_j \leq maxPre_{hj}$; so we know that

$$\sum_{j \in \hat{N}} (p_{ij} + Pre_j) \leq \sum_{j \in \hat{N}} (p_{ij} + maxPre_{hj}) = \sum_{j \in \hat{N}} \theta_{hij}.$$

Our assumption (17) therefore cannot hold as it is contradicted by

$$C_N \leq C_{\bar{N}} + \sum_{j \in \hat{N}} (p_{ij} + Pre_j) \leq C_{\bar{N}} + \sum_{j \in \hat{N}} \theta_{hij}$$

Therefore, we know that the cut provides a true lower bound on the achievable makespan of a schedule in future iterations. ▮

3.4 Stopping Condition

The Benders approach will iterate between master problem and subproblems until an optimal solution is found and proved. Optimality is proved if either of two conditions is met. The first condition is that all subproblems solved during an iteration find makespans less than or equal to the C_{max} value in the master problem. The second condition is that the C_{max} value found from the master problem is equal to the best feasible makespan found so far as defined in Section 3.2.

4 Computational Results

The Benders decomposition model and MIP model were tested on an AMD 270 CPU with 1 MB cache per core, 4 GB of main memory, running Red Hat Enterprise Linux 4. The MIP master problem was solved with IBM ILOG CPLEX 12.1 and the TSP solver used was tsp_solve.[2] Experiments were run for problem instances of between 10 and 60 jobs in increments of 10 jobs. For each job size, between 2 and 5 machines were tested. Each of these combinations have 10 instances for a total of 280 instances. A time limit of 3 hours was used for each instance. Processing times for each machine-job pair were generated from a uniform distribution between 5 and 200. To obtain setup times that were sequence dependent and follow the triangular inequality assumption, each job was given two different sets of coordinates on a Cartesian plane for every machine. The setup times are the Manhattan distances between two jobs' coordinates. Distances between the second set of coordinates are used to provide asymmetric setup times. We set our setup times to be between 25 and 50.

Table 1 shows results comparing MIP and Benders decomposition. For these results, the time until an optimal solution was found and proved were recorded. Where a time out occurred, 3 hours was used.

The Benders decomposition results are a significant improvement over the MIP performance. It is clear that the Benders decomposition approach is capable of solving much larger problems in significantly shorter run times. The Benders decomposition approach solves up to 60 jobs within the time limit. In contrast, the MIP model is able to solve only 10 jobs in the 3 hour time limit and is unable to solve all but 1 instance when there are 2 machines and 10 jobs.

[2] A TSP solver in C++ available online at http://www.or.deis.unibo.it/research pages/tspsoft.html.

		MIP		Benders	
N	M	Avg Runtime	# uns.	Avg Runtime	# uns.
10	2	9145.00	9	**0.07**	0
	3	4240.33	7	**0.10**	0
	4	1116.41	1	**0.18**	0
	5	11.48	0	**0.32**	0
20	2	10800.00	10	**0.23**	0
	3	10800.00	10	**0.62**	0
	4	10800.00	10	**2.80**	0
	5	10800.00	10	**9.44**	0
30	2	10800.00	10	**1.00**	0
	3	10800.00	10	**7.00**	0
	4	10800.00	10	**18.34**	0
	5	10800.00	10	**115.16**	0
40	2	10800.00	10	**1.21**	0
	3	10800.00	10	**12.96**	0
	4	10800.00	10	**74.75**	0
	5	10800.00	10	**411.65**	0
50	2	10800.00	10	**1.58**	0
	3	10800.00	10	**15.92**	0
	4	10800.00	10	**110.40**	0
	5	10800.00	10	**748.57**	0
60	2	10800.00	10	**1.95**	0
	3	10800.00	10	**88.30**	0
	4	10800.00	10	**544.25**	0
	5	10800.00	10	**3909.93**	0

Table 1. Comparison of MIP and Benders: Average CPU run-time in seconds and the number of unsolved instances.

Increasing the number of machines has a greater effect on the performance of the Benders models than increasing the number of jobs. The master problem had difficulty solving for increased machines. In fact, the TSP subproblem is able to solve each subproblem in milliseconds while the MIP master problem can spend over an hour searching for an assignment in a few cases. The opposite is seen for the MIP model. The MIP model has difficulties sequencing large number of jobs on machines and so, in the case where there are only 2 machines, the MIP model has a very high run-time for instances of 10 jobs and 2 machines. When the number of machines is increased to 5, the sequencing problem is simpler and results in fast run-times.

These results show that the Benders decomposition is able to optimally solve significantly larger problems than previously possible. The specialized branch-and-bound algorithm by Rocha et al. [22] discussed above could only solve up to 30 jobs with problems of similar parameters. When problem sizes are increased to 40 jobs, the branch-and-bound algorithm is found to provide solutions 1.69% worse than a heuristic model. At 60 jobs, the solutions found were 2.64% worse than the tested heuristic where the Benders decomposition is able to provide optimal solutions for these sizes. However, the branch-and-bound algorithm is solving a generalization of the PMSP problem we study and so a more comprehensive comparison is of interest.

5 Relaxed Benders Decomposition

Though the Benders decomposition approach obtains significant speed-ups, the problem sizes that can be solved are limited compared to what metaheuristic models can heuristically solve. As noted, instances of 120 jobs are tested in previous papers using heuristics and local search [15, 21]. Specifically, in work done by Helal et al. [15], schedules for problem instances of 100 jobs and 10 machines are found within minutes. The quality of these schedules is difficult to assess given that the optimal solutions are not known. However, for smaller instances (8 jobs and 4 machines) the optimal solution was known from the MIP model and the heuristic used was experimentally shown to have up to 5% deviation from optimal.

If optimal solutions are not possible for large instances, it is clear that heuristic solutions are necessary. Stopping the Benders program early and using the best found schedule as discussed in Section 3.2.1 is one approach. However, the problems must be small enough that the Benders model can solve one complete iteration in a reasonable time. From our experiments, we found that in some instances of 60 jobs and 5 machines, one complete iteration took more than one hour. Solve times of the subproblem are almost instantaneous, but the MIP master problem appears intractable once the size of the problem reaches 70 jobs and 5 machines. For problems as large as 100 jobs, the Benders decomposition model is not likely to solve the master problem within a reasonable time.

Therefore, we propose to find a sub-optimal master solution. During the search for a solution to the MIP, we allow the solver to stop once a solution is found within some predetermined gap from the best lower bound obtained. Doing so reduces the effort required in the master problem and enables the model to generate feasible schedules faster. The change to the Benders decomposition master problem results in it no longer being a true lower bound. However, it is guaranteed that the quality of the solution found is within the chosen optimality gap. Preliminary experiments found a gap of 5% to be a good trade off between run-time and quality for large problems. Thus, we make use of a 5% gap in the results we show here.

To compare the relaxed Benders decomposition against state-of-the-art heuristics, we run our model on the same test instances used to evaluate the meta-heuristic algorithms.[3] The performance of various meta-heuristics on these instances can be found at *Scheduling Research* [20]. In the above experiments, processing time and setup times were varied between 5 and 200, and 25 and 50 respectively. The instances used in the experiments have processing time and setup times ranging from 50 to 100. We vary the number of jobs evaluated between 40 and 120 with increments of 20 jobs. The number of machines used varies between 2 and 8 in increments of 2. However, machine-job pairs are only considered when the number of jobs are at least 15 times larger than the number of machines. For each tested machine-job pair, we test our model on 15 instances.

We directly compare the solution found from the relaxed Benders to a lower bound (LB) calculated for each problem instance. The lower bound used is the same as that used in the analysis of all heuristics in the work on ACO and is calculated using the equations

$$\begin{aligned} LB1 &= \tfrac{1}{|M|}\textstyle\sum_{j\in N}\min_{i\in M;k\in N}[p_{jk}+s_{ijk}] \\ LB2 &= \max_{j\in N}\{\min_{i\in M;k\in N}[p_{jk}+s_{ijk}]\} \\ LB &= \max(LB1, LB2) \end{aligned}$$

The deviation of the schedule found from the lower bound is $\Delta = \frac{C_{max}(*)-LB}{LB}$. Here, $C_{max}(*)$ represents the makespan found from using the specified scheduling algorithm.

Table 2 demonstrates that the Benders approach outperforms the state-of-the-art heuristics. The performance of ACO here is not the same as those presented in previously published work [4]. We use the results from an updated ACO model and compare against the performance of this model when looking at Δ since the new model produces better schedules on all instances. The results of the updated ACO algorithm are available on *Scheduling Research* [20].

The Benders model with a 5% gap performs best when the num-

[3] Thanks to Dr. Ghaith Rabadi for supplying the instances.

		Tabu Search	ACO		Benders 5%	
N	M	Δ	Δ	Runtime	Δ	Runtime
40	2	6.45	**2.15**	127.2	2.44	**0.24**
60	2	6.45	**1.57**	255.00	2.71	**0.31**
	4	8.17	**4.54**	192.60	4.74	**0.74**
80	2	5.95	**1.25**	416.40	1.66	**0.39**
	4	7.66	3.97	311.40	**3.77**	**9.21**
100	2	6.21	**1.08**	626.40	1.33	**6.99**
	4	7.06	3.54	557.40	**3.50**	**177.94**
	6	8.84	5.58	**544.20**	**5.42**	875.09
120	2	6.27	**0.92**	873.00	1.23	**15.05**
	4	6.80	3.00	727.20	**2.86**	**186.97**
	6	8.20	4.52	**753.00**	**4.43**	888.53
	8	10.09	5.70	**825.00**	**4.58**	2691.42

Table 2. Relaxed Benders Performance: Average run-time in seconds. The best quality solutions and run-times have been bolded.

ber of machines are relatively large. ACO is able to obtain better performance on two machine problems and the four machine problems with forty and sixty jobs. Not only does the relaxed Benders approach produce good schedules, it is able to do so in a notably short time. Tabu Search did not have run-time information for comparison so that information was not included. On every instance where ACO is able to find lower makespans than the Benders decomposition, the run-time for Benders is orders of magnitude faster than ACO. This indicates that a 5% gap is too conservative for these instances with few machines. A better quality schedule could be found by decreasing the gap and spending more time searching for better solutions. In fact, the problem instances where ACO finds a better solution are problem sizes that can be solved optimally by Benders decomposition. An improvement on the relaxed Benders decomposition is to autonomously adapt the % gap based on the difficulty of the problem. We intend to pursue such a model in our future work.

6 Conclusion

We presented a logic-based Benders decomposition approach to minimize the makespan of an unrelated parallel machine scheduling problem with sequence and machine dependent setup times. A MIP model was defined to assign jobs to machines and produce a lower bound on the makespan of the problem. A TSP solver was then used to find optimal schedules for each machine. Computational results indicate that the cooperation of MIP and TSP can effectively find optimal solutions. We are able to solve instances six times larger than what was previously possible using a MIP formulation in the literature and obtain optimal solutions on problems of the same size up to six orders of magnitude faster. Further, our model solves problems twice the size of specialized branch-and-bound approach designed for a similar problem. Additionally, a relaxation of our model is presented and results show that good solutions can be found in much shorter time and for larger instances if optimality is sacrificed. The relaxed Benders is compared to the state-of-the-art meta-heuristic approaches and shown to be best on many instances with much lower time and a guarantee of being within 5% of optimality.

REFERENCES

[1] A. Al-Salem, 'Scheduling to minimize makespan on unrelated parallel machines with sequence dependent setup times', *Engineering Journal of the University of Qatar*, **17**, 177–187, (2004).

[2] A. Allahverdi, J.N.D. Gupta, and T. Aldowaisan, 'A review of scheduling research involving setup considerations', *Omega*, **27**(2), 219–239, (1999).

[3] A.L. Arcus, 'A computer method of sequencing operations for assembly lines', *International Journal of Production Research*, **4**(4), 259–277, (1965).

[4] J.P. Arnaout, G. Rabadi, and R. Musa, 'A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times', *Journal of Intelligent Manufacturing*, **21**(6), 693–701, (2010).

[5] K.R. Baker, *Introduction to sequencing and scheduling*, John Wiley & Sons, 1974.

[6] R. Baldacci, M. Battarra, and D. Vigo, 'Routing a heterogeneous fleet of vehicles', *The vehicle routing problem: latest advances and new challenges*, 3–27, (2008).

[7] T. Bektas, 'Formulations and benders decomposition algorithms for multidepot salesmen problems with load balancing', *European Journal of Operational Research*, (2011).

[8] Y. Chu and Q. Xia, 'A hybrid algorithm for a class of resource constrained scheduling problems', in *Proceedings of the 2nd Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pp. 110–124. Springer, (2005).

[9] S. Dunstall and A. Wirth, 'Heuristic methods for the identical parallel machine flowtime problem with set-up times', *Computers & Operations Research*, **32**(9), 2479–2491, (2005).

[10] F. Focacci, P. Laborie, and W. Nuijten, 'Solving scheduling problems with setup times and alternative resources', in *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling*, (2000).

[11] P.M. França, M. Gendreau, G. Laporte, and F.M. Muller, 'A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times', *International Journal of Production Economics*, **43**(2-3), 79–89, (1996).

[12] CA Glass, CN Potts, and P. Shade, 'Unrelated parallel machine scheduling using local search', *Mathematical and Computer Modelling*, **20**(2), 41–52, (1994).

[13] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G.R. Kan, 'Optimization and approximation in deterministic sequencing and scheduling: A survey', *Annals of Discrete Mathematics*, **5**(2), 287–326, (1979).

[14] A. Guinet, 'Textile production systems: a succession of non-identical parallel processor shops', *The Journal of the Operational Research Society*, **42**(8), 655–671, (1991).

[15] M. Helal, G. Rabadi, and A. Al-Salem, 'A tabu search algorithm to minimize the makespan for the unrelated parallel machines scheduling problem with setup times', *International Journal of Operations Research*, **3**(3), 182–192, (2006).

[16] JN Hooker, 'Verifying logic circuits by Benders decomposition', *Principles and Practice of Constraint Programming: The Newport Papers, MIT Press, Cambridge, MA*, 267–288, (1995).

[17] J.N. Hooker, 'Planning and scheduling by logic-based benders decomposition', *OPERATIONS RESEARCH-BALTIMORE THEN LINTHICUM-*, **55**(3), 588, (2007).

[18] ME Kurz and RG Askin, 'Heuristic scheduling of parallel machines with sequence-dependent set-up times', *International Journal of Production Research*, **39**(16), 3747–3769, (2001).

[19] G. Lancia, 'Scheduling jobs with release dates and tails on two unrelated parallel machines to minimize the makespan', *European Journal of Operational Research*, **120**(2), 277–288, (2000).

[20] G. Rabadi. Scheduling research virtual center: http://schedulingresearch.com/, 2008.

[21] G. Rabadi, R.J. Moraga, and A. Al-Salem, 'Heuristics for the unrelated parallel machine scheduling problem with setup times', *Journal of Intelligent Manufacturing*, **17**(1), 85–97, (2006).

[22] P.L. Rocha, M.G. Ravetti, G.R. Mateus, and P.M. Pardalos, 'Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times', *Computers & Operations Research*, **35**(4), 1250–1264, (2008).

[23] C.A. Valle, L.C. Martinez, A.S. Da Cunha, and G.R. Mateus, 'Heuristic and exact algorithms for a min–max selective vehicle routing problem', *Computers & Operations Research*, **38**(7), 1054–1065, (2011).

Corrections to Experimental Results

Tony T. Tran and **J. Christopher Beck**
September 2012

The results printed at the ECAI 2012 conference proceedings [1] are errorneous due to a bug in our code. Corrected results are presented below for Tables 1 and 2 of the paper. The overall results are the same, but the gap used in the relaxed Benders decomposition for Table 2 has changed from 5% to 2% in order to account for the changes in runtime and performance.

		MIP		**Benders**	
N	M	Avg Runtime	# uns.	Avg Runtime	# uns.
10	2	9145.00	9	0.03	0
	3	4240.33	7	0.06	0
	4	1116.41	1	0.14	0
	5	11.48	0	0.026	0
20	2	10800.00	10	0.13	0
	3	10800.00	10	0.34	0
	4	10800.00	10	1.02	0
	5	10800.00	10	3.11	0
30	2	10800.00	10	0.28	0
	3	10800.00	10	1.68	0
	4	10800.00	10	12.97	0
	5	10800.00	10	36.63	0
40	2	10800.00	10	0.56	0
	3	10800.00	10	3.84	0
	4	10800.00	10	38.75	0
	5	10800.00	10	152.00	0
50	2	10800.00	10	0.79	0
	3	10800.00	10	24.08	0
	4	10800.00	10	146.37	0
	5	10800.00	10	1026.37	0
60	2	10800.00	10	2.33	0
	3	10800.00	10	80.80	0
	4	10800.00	10	390.51	0
	5	10800.00	10	4613.74	3

Table 1: Comparison of MIP and Benders: Average CPU run-time in seconds and the number of unsolved instances.

		Tabu Search	ACO		Benders 2%	
N	M	Δ	Δ	Runtime	Δ	Runtime
40	2	6.45	**2.15**	127.2	2.47	**0.33**
60	2	6.45	**1.57**	255.00	2.40	**0.53**
	4	8.17	4.54	192.60	**4.13**	**7.18**
80	2	5.95	**1.25**	416.40	1.72	**1.11**
	4	7.66	3.97	311.40	**3.66**	**28.86**
100	2	6.21	**1.08**	626.40	1.28	**8.63**
	4	7.06	3.54	557.40	**3.24**	**212.75**
	6	8.84	5.58	**544.20**	**4.35**	2379.10
120	2	6.27	**0.92**	873.00	1.28	**17.73**
	4	6.80	3.00	727.20	**2.98**	**208.53**
	6	8.20	4.52	**753.00**	**3.37**	1187.55
	8	10.09	5.70	**825.00**	**4.28**	4746.82

Table 2: Relaxed Benders Performance: Average run-time in seconds. The best quality solutions and run-times have been bolded.

References

[1] T. T. Tran and J. C. Beck. Logic-based benders decomposition for alternative resource scheduling with sequence dependent setups. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012)*, pages 774–779, 2012.