Decomposition Techniques for Multistage Scheduling Problems Using Mixed-Integer and Constraint Programming Methods

Iiro Harjunkoski and Ignacio E. Grossmann*

Department of Chemical Engineering
Carnegie Mellon University, Pittsburgh, PA 15213

April 2002

Abstract

In this paper two strategies are presented to reduce the combinatorial complexity when solving single stage and multistage optimization scheduling problems that involve cost minimization and due dates. These problems can naturally be decomposed into assignment and sequencing subproblems. The proposed strategies rely on either combining mixed-integer programming (MILP) to model the assignment part and constraint programming (CP) for modeling the sequencing part, or else combining MILP models for both parts. The subproblems are solved sequentially by adding integer cuts to the first MILP to generate new assignments. Results are presented for both single and multistage systems.

Keywords: Mixed Integer Programming, Constraint Programming, Hybrid Strategy, Multistage Scheduling.

1. INTRODUCTION

With increased computational power and algorithmic developments Mixed-Integer Linear Programming (MILP) has become a competitive alternative for solving scheduling and related combinatorial optimization problems in chemical engineering. Examples of MILP-models for scheduling are given, for instance, in Pekny and Reklaitis (1998), Shah (1998) and Pinto and Grossmann (1998). The advantage of an MILP-approach is that it provides a general framework for modeling a large variety of problems such as multiperiod planning, job shop scheduling and supply chain management problems. However, the major difficulty lies in the computational expense that may be involved in solving large-scale problems, which is due to the computational complexity of MILP problems which are NP-hard. Therefore, solving MILP models without simplification through the use of heuristics can often make this approach prohibitive

* Author to whom all correspondence should be addressed. Email: grossmann@cmu.edu

for industrial applications. To reduce the combinatorial explosion in scheduling problems, logic based optimization methods have emerged (Hooker, 2000). These include Generalized Disjunctive Programming (GDP) (Raman and Grossmann, 1994) and Constraint Programming (CP) (Hentenryck, 1989). GDP is based on the idea of representing discrete and continuous optimization problems through equations, disjunctions and logic propositions, and in terms of Boolean and continuous variables. CP is similar to GDP in that it also involves equations, logic statements and disjunctions. However, the most important difference is that CP has high-level constructs known as global constraints that are procedural and implicit in nature, which can make it possible to express the models in a very compact form. In contrast, GDP gives rise to declarative models that are expressed through explicit equations. Furthermore, while GDP problems can be solved through branch and bound, or reformulated as MILPs, CPs are solved through implicit enumeration techniques coupled with techniques for domain reduction of variables, which in turn are based on constraint propagation techniques. In the case of scheduling, these correspond for instance to edge finding methods (Carlier and Pinson, 1989) that have proved to be efficient in solving certain types of jobshop scheduling problems.

Software for MILP has been available for over two decades. Current codes include OSL (IBM, 1992), ILOG CPLEX (ILOG, 2001) and XPRESS-MP (Dash Associates, 1999). For logic based optimization methods software has emerged only in recent years. Perhaps the most popular software for solving CPs has been ILOG Solver (ILOG, 1999), ECLiPSe (Wallace et al., 1997) and CHIP (Dincbas et al., 1988). While for general purpose problems ILOG Solver is often less effective than MILP, the special extension for scheduling, ILOG Scheduler, has proved to be quite effective, particularly for discrete manufacturing scheduling problems where feasibility is a major issue. Domain reduction techniques perform well for these problems since they have finite or discrete domains.

Interest for combining MILP and CP has arisen recently (Ottosson et al., 2001), because these methods have proved to be successful in solving complementary classes of problems: MILP when optimization is the dominating aspect, and CP when feasibility is the major concern. The main objective of developing hybrid MILP/CP strategies is to try to use their complementary strengths in an efficient manner. A major goal of this paper is to classify and further develop hybrid methods for solving scheduling problems, as well as bringing up some issues that have until now been discussed only in the OR-literature. In the following section, we will review some of the recent developments made for the scheduling models. We then describe two decomposition strategies that either integrate MILP and CP, or are composed of two MILP subproblems. The application of the strategies is illustrated in the single stage batch plant, and in multiple stage batch plants. The former problem has been addressed before by Jain and Grossmann (2001), while the latter is a new application for hybrid models in which the generation of cuts is significantly more difficult.

2. BACKGROUND

There has been considerable research on methods for solving MILP problems (Nemhauser and Wolsey, 1988). Recently the most significant development has been in new software implementations that can increasingly solve larger problems (e.g. CPLEX, XPRESS-MP). Most of these apply the Branch and Bound method for handling the integer variables, in which the greatest bottleneck arises from the potentially large integer search space due to the exponential number of possible combinations. The effectiveness of MILP methods depends further on the size of

the linear programming (LP) subproblems, and on the gap between the objective value for the optimal solution and the initial LP subproblem. General approaches to overcome this bottleneck include preprocessing and Branch and Cut algorithms (Johnson et al., 2000) in which extra cuts are added to the MILP to strengthen its relaxation gap, as for instance the lift and project cuts by Balas et al. (1996).

Constraint programming applies constraint propagation at every node for reducing the domain of all the variables (Hentenryck, 1989; Marriott and Stuckey, 1999). The variable domains may be continuous, discrete or boolean. An empty domain means that the node is infeasible and can be fathomed. Branching is performed on domains that contain more than one element (variable value). CP was originally developed for solving feasibility problems, but has also been extended to solve optimization problems by placing the objective function in an inequality constraint that is less than or equal to the incumbent solution. The search terminates when the domain of this constraint is empty and all nodes are fathomed. It should be noted that although CP methods are often efficient in solving discrete feasibility problems, the performance in optimization depends highly on the propagation algorithms used and on how constrained the problem is.

In MILP all the constraints are evaluated simultaneously, while CP evaluates the effect of constraints sequentially by communicating through variable domains. Consequently, it is generally difficult to obtain the optimal solution for loosely constrained CP problems. On the other hand, MILP methods require all constraints to be linear equalities or inequalities due to the LP based model. This restriction does not apply for CP problems, in which all algebraic constructs are allowed together with some special constructs, known as global constraints, such as "all different" or "cumulative" constraints (ILOG, 1999).

Apart from the algorithmic development for improving the solution of multistage scheduling problems, a number of different modeling approaches have been presented in the chemical engineering literature. The general idea of these is mostly to split the large problem into smaller units or to apply some supporting heuristics. Examples of this can be seen for instance in Pinto and Grossmann (1995), who used a slot-based MILP scheduling model combined with single machine scheduling algorithms. In contrast, in Méndez et al. (2001) the variables of a multistage flowshop problem are split into assignment and sequencing, in a similar fashion as in Ierapetritou and Floudas (1998) for the STN model, in order to reduce the number if 0-1 variables. Wang and Zheng (2001) presented a hybrid approach where the authors combine simulated annealing and genetic algorithm for a job-shop scheduling problem in a parallel framework. When trying to solve large-scale problems, the former strategies suffer from combinatorial explosion and weak relaxations due to the big-M terms, while the latter cannot guarantee the solution quality.

3. DECOMPOSITION STRATEGIES

We assume in this paper that we deal with general scheduling problems that involve cost minimization with due dates and sequence independent setup times. Assuming this type of problems motivated by the work of Jain and Grossmann (2001), we consider decomposition strategies for scheduling into assignment and sequencing levels as shown in Fig. 1. As can be seen, the main idea is to determine optimal assignments of jobs to units or machines at the high level, and then perform feasible sequencing of the jobs for the given assignments at a lower level. Since the latter may not give rise to a feasible sequence, cuts are derived and added to

the assignment problem to generate a new assignment. We explore in this paper the cases when an MILP is used for the assignment subproblem which is driven by optimization, and either CP or MILP is used to solve the sequencing subproblem, which is dominated by feasibility concerns. These strategies are applied to both single- and multistage scheduling problems.

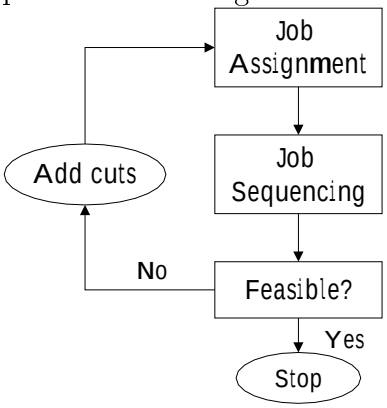

Figure 1. Decomposition strategy

A review of different integration strategies has been reported in Jain and Grossmann (2001). These authors proposed an integration strategy using a hybrid MILP/CP model consisting of two subproblems, as shown in Fig. 1. One subproblem deals with the assignment of orders to units which is modeled as an MILP. The other subproblem deals with the sequencing of orders for a fixed assignment and it is modeled as a CP problem. These subproblems are integrated in a decomposition strategy in which the MILP is solved as a master problem relaxing the sequencing constraints, while the CP is solved as a feasibility subproblem for the sequencing.

Consider a scheduling problem where a complete MILP model can be expressed as follows.

$$min \qquad c_u^T u + c_x^T x \tag{1}$$

subject to

$$Au + Bx + Dv + Ey \leq a \tag{2}$$

$$A'u + B'x + D'v + E'y \leq a' \tag{3}$$

$$u \in R^u, v \in R^v, x \in Z^x, y \in Z^y$$

The problem has both continuous (u, v) and discrete variables (x, y), and assumes that only the variables (u, x) are present in the objective function. It is also assumed that the MILP problem can be divided into two parts. One part is suitable for the MILP framework, i.e. the objective function and constraints (2) give rise to tight LP-relaxations. The other part leads to a feasibility problem given by constraints (3) that exhibit poor LP-relaxation. The former variables and constraints form the core of the MILP subproblem, and the latter ones are left either for the CP subproblem, or for a MILP with fixed assignments. The main difference compared to earlier approaches is that neither of the subproblems are given in complete form as MILP or CP problems.

If the same problem were to be modeled as a CP, more constructs can be utilized and thus some variable definitions may be different, but equivalence relations can be established between these two models. The corresponding CP model is as follows,

$$min \quad f(\bar{u}, \bar{x}) \tag{4}$$

subject to

$$G(\bar{u}, \bar{x}, \bar{v}, \bar{y}) \leq 0 \tag{5}$$

$$\bar{u}, \bar{x}, \bar{v}, \bar{y} \in D$$

Here, the $\bar{u}, \bar{x}, \bar{v}, \bar{y}$ are the corresponding CP variables and D is their domain. Typically the constraint set generated for the CP is much smaller than that for the MILP, but the CP may have difficulties in finding a solution and proving the optimality, mainly due to the lack of linear programming relaxations.

In Strategy I, which is the one that was proposed by Jain and Grossmann (2001), we consider a hybrid model where the complicating MILP constraints in Eq. (3) are replaced by their CP equivalent, $\bar{G}(\bar{u},\ \bar{x},\ \bar{v},\ \bar{y}) \leq 0$ for fixed $(\bar{u},\ \bar{x})$.

$$min \qquad c_u^T u + c_x^T x \tag{6}$$

subject to

$$Au + Bx + Dv + Ey \leq a \tag{7}$$

$$(u, x) \Leftrightarrow (\bar{u},\ \bar{x}) \tag{8}$$

$$\bar{G}(\bar{u},\ \bar{x},\ \bar{v},\ \bar{y}) \leq 0 \tag{9}$$

$$u \in R^u, v \in R^v, x \in Z^x, y \in Z^y$$

$$\bar{u},\ \bar{x},\ \bar{v},\ \bar{y} \in \mathrm{D}$$

Note that the objective function (6) is exactly the same as for the MILP model, and that the equivalence in (8) relates the MILP and CP variables. The variables (v, y) and $(\bar{v}, \bar{y})$ may or may not be directly connected, depending on the specific problem. The MILP subproblem is then given by (6), (7) and integer cuts that are added at each major iteration, while the CP subproblem is given by (9) for fixed values of $\bar{u}, \bar{x}$. By solving the relaxed MILP problem, we obtain an optimal solution that satisfies the constraint sets in (7). This solution can be used to derive a partial CP solution through the equivalence relations (8) and the feasibility problem for fixed $(u,\ x)$ will verify if a solution satisfying the constraints (9) can be found. If this is the case, then the optimal solution is found and the hybrid algorithm will terminate. Else, if the CP solution is infeasible, then as many infeasible solutions as possible need to be eliminated by cuts that are added to the next relaxed MILP problem.

In Strategy II, the required MILP model (1)-(3) is used except that it is solved by decomposing it into two levels: (a) an assignment part consisting of (1) and (2) with integer cuts, and a sequencing part consisting of the constraints in (3) for fixed u and x. In other words, the only difference with Strategy I is that instead of solving a CP subproblem for the sequencing part, we solve a feasibility MILP problem. Although the MILP sequencing problem has in general poor relaxation, the advantage is that it is simplified and reduced in size when fixing the job assignments. Furthermore, one can resort to the same solver in both phases.

The following sections of this paper apply the two strategies to single stage and multistage scheduling problems. The former was studied previously by Jain and Grossmann (2001), while the latter has not been addressed before with a hybrid decomposition strategy.

4. SINGLE-STAGE PARALLEL SCHEDULING PROBLEM

4.1. Strategy I

Strategy I of the previous section was proposed by Jain and Grossmann (2001) and applied to a single-stage scheduling problem for parallel dissimilar machines. The approach reduced dramatically the computation time and reruns with expanded data sets in Harjunkoski et al. (2000) demonstrated the robustness of the method. The main idea lies in that the assignment of orders to equipment is performed with a relaxed MILP master model, while the sequencing is performed with a CP model. This is motivated by the fact that assignment problems tend to solve well with MILP and using CP for the sequencing allows the usage of special scheduling constructs that may improve the solution efficiency. The decomposition is demonstrated in Fig. 2.

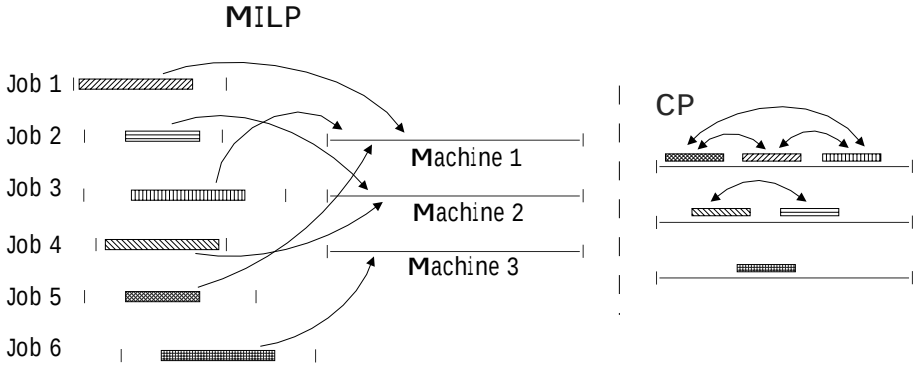

Figure 2. MILP/CP decomposition for a single-stage problem

The resulting hybrid MILP/CP model reported in Jain and Grossmann (2001) is given below. Eqs. (10)-(14), (20) and (21) form the MILP assignment problem and the CP sequencing problem is defined by Eqs. (15)-(19) and (22)-(24).

$$min \qquad \sum_{i \in I} \sum_{m \in M} C_{im} y_{im} \tag{10}$$

subject to

$$ts_i \geq r_i \quad \forall i \in I \tag{11}$$

$$ts_i \leq d_i - \sum_{m \in M} p_{im} y_{im} \quad \forall i \in I \tag{12}$$

$$\sum_{m \in M} y_{im} = 1 \quad \forall i \in I \tag{13}$$

$$\sum_{i \in I} y_{im} p_{im} \leq \max_i \{d_i\} - \min_i \{r_i\} \quad \forall m \in M \tag{14}$$

$$\text{if } (y_{im} = 1) \text{ then } (z_i = m) \quad \forall i \in I, \ \forall m \in M \tag{15}$$

$$i.\text{start} \geq r_i \quad \forall i \in I \tag{16}$$

$$i.\text{start} \leq d_i - p_{z_i} \quad \forall i \in I \tag{17}$$

$$i.\text{duration} = p_{z_i} \quad \forall i \in I \tag{18}$$

$$i \text{ requires } q_{z_i} \quad \forall i \in I \tag{19}$$

$$ts_i \geq 0 \tag{20}$$

$$y_{im} \in \{0,1\} \quad \forall i \in I,\ \forall m \in M \tag{21}$$

$$z_i \in M \quad \forall i \in I \tag{22}$$

$$i.\text{start} \in Z \quad \forall i \in I \tag{23}$$

$$i.\text{duration} \in Z \quad \forall i \in I \tag{24}$$

In the above model, I is the set of jobs and M is the set of machines. The assignment is represented by the binary variable y_{im} and the cost of each assignment is given by C_{im}. Thus, the objective function (10) minimizes the total assignment or processing cost. Eq. (11) states that the starting time, ts_i, of a job should be greater than its release date, r_i. Similarly, Eq. (12) defines that job i should be started before its due date, d_i, subtracted with the required production time, p_{im}, on the assigned machine. These constraints reduce the search space. Constraint (13) ensures that each job is assigned to exactly one machine. The search space is further reduced by Eq. (14) that limits the total production time on each machine to less than the time between the latest due date and earliest release date. The only variables of the MILP problem are the binary assignment variables and the positive starting times. The link between the MILP and CP subproblems is defined by Eq. (15) where the assignment is transformed to the CP problem through variable subscripts. In the sequencing problem the start time is defined correspondingly with Eqs. (16)-(17), and the duration is defined in Eq. (18). Note that the i.duration and i.start are built-in variables in OPL. Constraint (19) utilizes a special ILOG Scheduler construct 'requires' and states that each job i needs resources from one of the machine pools q. The model ends with the variable definitions in Eqs. (21)-(24).

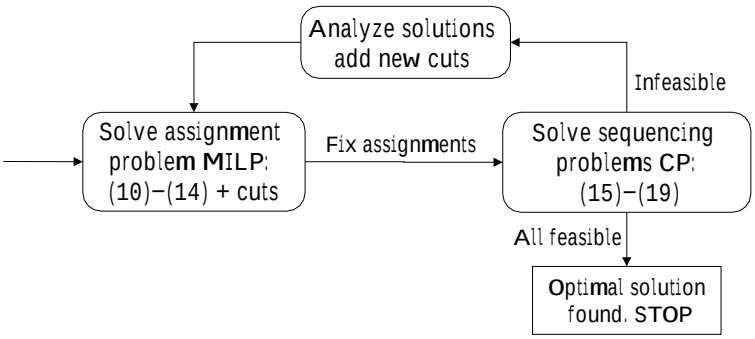

Figure 3. Hybrid Strategy I

The iterative Strategy I, whose convergence was proved by Jain and Grossmann (2001), is presented in Fig. 3. Since the problem is a single-stage scheduling problem, it should be noted that the sequencing subproblems can be solved separately for each machine since they become decoupled when fixing the job assignments. Solutions of the CP problems can be used to generate cuts that are incorporated to the MILP master problem. If one of the assignments cannot be sequenced, the assignment is infeasible and the cut that eliminates this assignment from the next MILP can be expressed as,

$$\sum_{i \in I} a_{im} \cdot y_{im} \leq \sum_{i \in I} a_{im} - 1 \qquad \forall m \in M \tag{25}$$

where a_{im} are 0-1 coefficients for the assignment variables y_{im}. As an example, if all the assignments in Fig. 2 were to be infeasible, the cuts would be as follows:

$$y_{11} + y_{31} + y_{51} \leq 2$$

$$y_{22} + y_{42} \leq 1$$

$$y_{63} \leq 0$$

Note that these cuts also eliminate all supersets of the assignments. The alternating MILP and CP problems are solved until all sequencing problems result in feasible solutions as shown in Fig. 3. It should be pointed out that if no solution can be obtained for the relaxed MILP problem, then the original problem is also infeasible. A detailed description of the method can be found in Jain and Grossmann (2001). Five scheduling problems, each with two new data sets that are more complex than the ones in Jain and Grossmann (2001), given in Appendix A, have been solved using OPL-Studio 3.5 on a 933MHz Pentium III computer running Redhat Linux. The results are shown in Table 1.

Table 1. Single-stage scheduling results

Problem,	Obj	CPU-s			Details of hybrid method			
data	value	MILP	CP	Hybrid*	Iter	Cuts	sub-MILP	sub-CP
1,1	26	0.01	0.03	0.32	2	1	0.00	0.00
1,2	21	0.02	0.01	0.34	1	0	0.00	0.00
2,1	60	0.41	0.01	0.77	14	17	0.26	0.06
2,2	46	0.15	0.03	0.36	1	0	0.00	0.00
3,1	104	56.93	1.81	5.67	24	37	3.77	0.06
3,2	85	5.53	0.22	0.28	1	0	0.00	0.00
4,1	116	347.97	277.64	2.17	14	22	1.32	0.10
4,2	105	136.57	17.25	1.08	15	15	0.49	0.10
5,1	159	>130000[1]	189244.33	10.04	29	64	8.40	0.17
5,2	144	>130000[2]	5925.66	2.89	26	35	2.09	0.26

*) CPU-time also includes infeasible problems and overheads (problem loading, cut generation)

1) Best solution obtained at this point: 160

2) Optimal solution found but not verified

In Table 1 we first show the solution times of the complete MILP and CP formulations, followed by the total CPU-time needed by the hybrid strategy. The last four columns give the number of iterations and cuts used by the hybrid strategy and how big a part of the solution time was spent in solving the MILP assignment and CP sequencing subproblems. Note that the CPU-time reported for the CP subproblems is a sum of the feasible problems only, since ILOG Solver and Scheduler do not give any solution information on infeasible problems. Therefore, the difference between the total time and the sum of sub-MILP and sub-CP times is caused mainly by infeasible sequencing problems, and to a smaller extent by overheads such as problem

loading and the generation of cuts. It should also be stressed that the hybrid Strategy I is guaranteed to yield the same optimal solution as the MILP and CP methods.

4.2. Strategy II

We consider in this section an MILP-MILP Strategy II, where instead of solving one large MILP problem we solve two smaller MILP subproblems iteratively. The assignment problem is given by equations (10)-(14), while the MILP feasibility problem is for sequencing the jobs, i, that have been assigned to machine m', here declared as set I'. Let the variable $x_{ii'}$ be the sequencing variable, which is equal to one if job i precedes job i' and else zero. The MILP sequencing model is then as follows,

$$min \;\; 0 \tag{26}$$

subject to

$$ts_i \geq r_i \quad \forall i \in I' \tag{27}$$

$$ts_i \leq d_i - p_{im'} \quad \forall i \in I' \tag{28}$$

$$x_{ii'} + x_{i'i} = 1 \quad \forall (i,i') \in I', i > i' \tag{29}$$

$$ts_{i'} \geq ts_i + p_{im'} - \underset{i \in I'}{max}\{d_i\} \cdot (1 - x_{ii'}) \quad \forall (i,i') \in I', i \neq i' \tag{30}$$

$$x_{ii'} \in \{0,1\} \qquad ts_i > 0$$

The objective function is simply a "dummy" value of zero to define a feasibility problem (i.e. all feasible points have zero objective value). Constraints (27)-(28) are as before with the exception that here we only consider one machine, m'. Only one of the two alternative sequences is valid and this is stated in Eq. (29), followed by the final constraint (30), that defines the starting times, using the latest due date as the big-M parameter. It should be noted that in the implementation it is convenient to define the objective function as $min \sum_i \sum_{i'} x_{ii'}$ in order to expedite the branch and bound search since this objective will tend to take the discrete variables out of the basis.

In Strategy II, the assignment problem is exactly the same as before. Also, the number of iterations and cuts should be almost the same, since we still solve a feasibility problem, only now using MILP. Since the sequencing problems are significantly smaller than the complete MILP formulation this approach should also simplify the sequencing part of the problem.

Table 2. Single-stage scheduling results with the hybrid MILP-MILP strategy

Problem, data	Obj value	Total* CPU-s	Iter	Cuts	Details of hybrid method Assign	Sequence
1,1	26	0.43	2	1	0.01	0.01
1,2	21	0.32	1	0	0.00	0.00
2,1	60	1.13	13	15	0.36	0.21
2,2	46	0.42	1	0	0.00	0.03
3,1	104	12.71	23	38	7.53	0.85
3,2	85	0.52	1	0	0.01	0.12
4,1	116	5.05	12	20	2.08	1.40
4,2	105	4.32	15	15	0.23	0.88
5,1	159	41.54	28	63	23.53	7.32
5,2	144	162.00	27	35	4.22	10.90

*) CPU-time also includes infeasible problems and overheads (problem loading, cut generation)

Table 2. shows the results for the same test problems. The results show a significant improvement compared to the stand-alone MILP and CP models (see Table 1). Clearly the largest benefit is obtained through the decomposition of the original problem. Even if both Strategy I and Strategy II perform well, it needs to be pointed out that the total times that also include the time spent on solving infeasible sequencing problems, are very different. In general, MILP spends much more time on verifying infeasibility. In Strategy I, the infeasible subproblems do not require more than 35% of the total CPU-time in any of the example problems, but detecting infeasibility in Strategy II typically takes 50% of the total time. In problem 5,2 this takes more than 90% of the CPU-time and the performance is an order of magnitude worse than with Strategy I. Consequently, even if the hybrid MILP-CP and MILP-MILP approaches both result in significant improvements, CP seems to be more suitable for feasibility sequencing problems.

5. MULTISTAGE SCHEDULING PROBLEM

The final part of this paper focuses on a new application of the decomposition strategies to the multistage case. As will be seen, a major difference is that the cuts are considerably more complex to derive. We consider a variation of the multistage batch scheduling problem by Pinto and Grossmann (1995). The problem involves a multi-stage batch plant with dissimilar parallel equipment in each stage that must process a given number of orders that have specified due dates and that are to be processed at each stage. The chosen objective is to minimize the assignment or processing costs of the orders. The MILP optimization of this jobshop problem is often very expensive to solve. An example of the problem type is shown in Fig. 4.

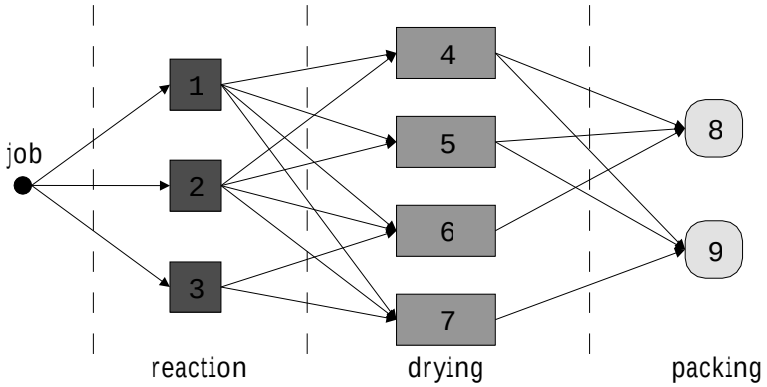

Figure 4. Multistage scheduling problem

The problem may contain some topological restrictions. Some units may not be able to process all orders. Furthermore, some production paths may be restricted as shown in Fig. 4, where, for instance, unit 7 can only be followed by unit 9.

We propose first a hybrid MILP/CP model for solving the multistage scheduling problem with Strategy I. The problem is decomposed into two subproblems: assignment problem and sequencing problem. In the case of single-stage scheduling, it was possible to solve each sequencing problem separately and an infeasible CP problem indicated that the assignment was invalid. In the multistage case this is unfortunately not possible, since the stages and machines are connected. Consequently, there are two main complications of which the first one is that the sequencing problem needs to be solved simultaneously for every job and machine, which results in large CP subproblems. The second complication comes from the fact that infeasible CPs do not provide useful information about the reason for the infeasibility. Therefore, the CP will be relaxed to minimize infeasibility (violation of due dates) to yield better cuts. As for Strategy II, using a similar treatment as in section 4.2, we replace the CP subproblem for sequencing by an equivalent MILP subproblem.

The multistage scheduling problem can, of course, be expressed either as a pure MILP or CP problem. We will first present these models and then discuss their possible strengths and weaknesses.

5.1. MILP model

In the following MILP model, the set I corresponds to jobs, set M refers to machines and K is the set of stages. The assignment variables, y_{im}, are 0-1 variables relating job i to machine m. The sequencing variables, $x_{ii'k}$, are equal to one if product i precedes product i' in stage k, else zero. In the models below, coefficients are given in uppercase and variables in lowercase letters.

$$min \quad \sum_{i \in I} \sum_{m \in M} C'_{im} \cdot y_{im} + \sum_{m \in M} C''_{m} \cdot z_m \tag{31}$$

subject to

$$\sum_{m \in M(k)} y_{im} = 1 \qquad \forall i \in I,\ \forall k \in K \tag{32}$$

$$y_{im} \leq z_m \qquad \forall i \in I,\ \forall m \in M \tag{33}$$

$$c_{ik} \leq T_i^d \qquad \forall i \in I, k = |K| \tag{34}$$

$$c_{ik} \geq \sum_{m \in M(k)} y_{im} \cdot (T_i^r + T_m^s + T_{im}^p) \qquad \forall i \in I,\ \forall k \in K \tag{35}$$

$$c_{ik} \leq c_{i,k+1} - \sum_{m \in M(k+1)} y_{im} \cdot (T_{im}^p + T_m^s) \qquad \forall i \in I, k < |K| \tag{36}$$

$$c_{i'k} \geq c_{ik} + T_{i'm}^p + T_m^s - \mathrm{U} \cdot (1 - x_{ii'k} + 1 - y_{im} + 1 - y_{i'm})$$
$$\forall i, i' \in I, i < i',\ \forall k \in K,\ \forall m \in M(k) \tag{37}$$

$$c_{ik} \geq c_{i'k} + T_{im}^p + T_m^s - \mathrm{U} \cdot (x_{ii'k} + 1 - y_{im} + 1 - y_{i'm})$$
$$\forall i, i' \in I, i < i',\ \forall k \in K,\ \forall m \in M(k) \tag{38}$$

$$y_{im} = 0 \qquad \forall i \in I,\ \forall m \in M, (i,m) \in \hat{B} \tag{39}$$

$$y_{im_1} + y_{im_2} \leq 1 \qquad \forall i \in I,\ \forall (m_1, m_2) \in \hat{M} \tag{40}$$

$$\sum_{i \in I} y_{im} \cdot (T_{im}^p + T_m^s) \leq \max_{i \in I} \left\{ T_i^d - \sum_{k' \in K, k' > k} \min_{m' \in M(k'), (i,m') \notin \hat{B}} (T_{im'}^p + T_{m'}^s) \right\} -$$
$$\min_{i \in I} \left\{ T_i^r + \sum_{k' \in K, k' < k} \min_{m' \in M(k'), (i,m') \notin \hat{B}} (T_{im'}^p + T_{m'}^s) \right\} \qquad \forall k \in K,\ \forall m \in M(k) \tag{41}$$

$$y_{im}, x_{ii'k} \in \{0,1\} \qquad z_m, c_{ik} \geq 0$$

The objective function (31) minimizes the assignment costs. The coefficient, C'_{im}, is a general assignment cost that may be associated, for instance, with the processing time, while the second coefficient, C''_m, represents a one-time cost, such as initialization or startup costs for a specific unit. Here we apply a fixed value $C''_m = 10$ for all problems, even though the main idea is to avoid the activation of an extra unit if this only leads to a minimal saving. The assignment constraint (32) states that each job i has to be processed on exactly one unit m at every stage k. Machines of a certain stage k belong to the set $M(k)$. Equation (33) states that units that are in use also need to be initialized. The finishing time, c_{ik}, of job i in stage k is defined in Eqs. (34)-(36), which enforce it to be less than the due date (T_i^d), greater than the release date (T_i^r) added with a machine-dependent setup time, (T_m^s) and the processing (T_{im}^p) time and, finally, less than the starting time of the next stage. The sequencing is performed by the constraints (37)-(38), of which only one may be true. This is enforced by the big-M terms including the assignment and sequencing variables. The three last constraints will reduce the search space. Forbidden assignments are specified in Eq. (39), where $\hat{B}$ is a set of forbidden (job,machine) combinations. Similarly, constraint (40) prohibits any job i to go from machine m_1 to m_2, if these are part of the set of non-existing processing paths, $\hat{M}$. Constraint (41) reduces the search domain by making sure that the total processing time of the jobs on a machine will fit between 1) the maximum due date subtracted with the shortest processing and setup times of all later stages, and 2) the minimum release date plus the shortest processing times of all earlier stages. This is illustrated in Figures 5a and 5b.

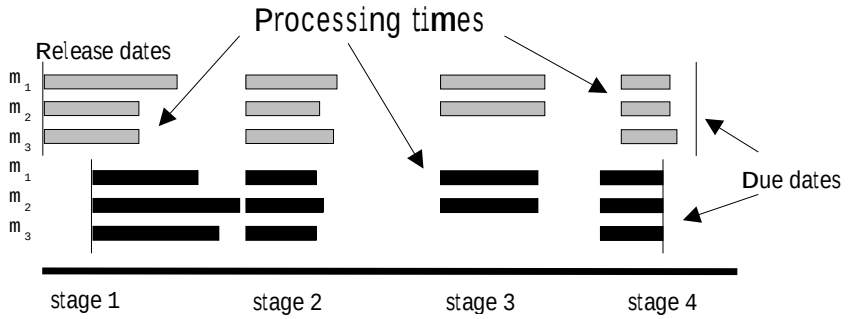

Figure 5a. Processing times for two jobs in 11 machines

The time domain can be reduced significantly by using constraint (41), which may eliminate a large number of invalid assignments. In Figure 5a, there are two products (black and gray) and their corresponding processing times on each unit. Notice that stage 3 contains only two parallel units, while the rest of the stages have three parallel units. The constraint is applied on stage 3 and the result is shown in Figure 5b.

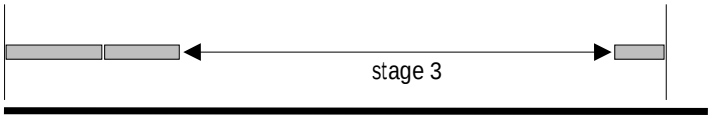

Figure 5b. Available timeframe for stage 3

Although the MILP model represents exactly the given problem, it has the drawback that the number of binary variables is relatively large mainly due to the three-indexed variables, which may lead to a combinatorial explosion. Also, the relaxations of the sequencing constraints are very weak and this formulation is intractable for solving large scheduling problems. To try to overcome this problem one can decompose the MILP using Strategy II in which we solve subproblems given by (31)-(35) and (39)-(41) for the assignment part and the constraints in (36)-(38) as a feasibility MILP for the sequencing part. This sequencing subproblem can also be modeled as a CP problem using Strategy I.

5.2. CP model

The complete multistage scheduling problem can be modeled as a CP problem as shown in this section. The constraint programming model may benefit from a large variety of special constructs designed for solving scheduling problems. Therefore, the CP model differs clearly from the corresponding MILP representation and is given below. Here, the syntax is based on OPL (ILOG, 2001) and specific CP names for global constraints,

$$min \quad \sum_{i \in I} \sum_{m \in M} C'_{im} \cdot y_{im} + \sum_{m \in M} C''_{m} \cdot z_{m} \tag{42}$$

subject to

$$Stage[i,1].\text{start} \geq T_i^r \qquad \forall i \in I \tag{43}$$

$$Stage[i,|K|].\text{end} \leq T_i^d \qquad \forall i \in I \tag{44}$$

$$Stage[i,k] \text{ requires } s \qquad \forall i \in I,\ \forall k \in K \tag{45}$$

$$\text{activityHasSelectedResource}(Stage[i,k], s, tool[m]) \Rightarrow dur[i,k] = T_{im}^p + T_m^s$$
$$\forall i \in I,\ \forall k \in K,\ \forall m \in M \tag{46}$$

$$\text{not activityHasSelectedResource}(Stage[i,k], s, tool[m])$$

$$\forall i \in I,\ \forall k \in K,\ \forall m \in M | m \notin M(k) \tag{47}$$

$$\text{not activityHasSelectedResource}(Stage[i,k], s, tool[m])$$

$$\forall i \in I,\ \forall k \in K,\ \forall m \in M(k) | (i,m) \in \hat{B} \tag{48}$$

$$\text{activityHasSelectedResource}(Stage[i,k], s, tool[m_1]) \Rightarrow$$

$$\text{not activityHasSelectedResource}(Stage[i, \text{next}(k)], s, tool[m_2])$$

$$\forall i \in I,\ \forall k \in K,\ \forall m_1, m_2 \in M \mid k < |K|, (m_1, m_2) \in \hat{M} \tag{49}$$

$$Stage[i,k] \text{ precedes } Stage[i, \text{next}(k)] \qquad \forall i \in I,\ \forall k \in K \mid k < |K| \tag{50}$$

$$\text{activityHasSelectedResource}(Stage[i,k], s, tool[m]) \Rightarrow y_{im} = 1, z_m = 1$$

$$\forall i \in I,\ \forall k \in K,\ \forall m \in M \tag{51}$$

$$\min_{m \in M(k) | (i,m) \notin \hat{B}} \{T^p_{im} + T^s_m\} \leq dur[i,k] \leq \max_{m \in M(k) | (i,m) \notin \hat{B}} \{T^p_{im} + T^s_m\} \qquad \forall i \in I,\ \forall k \in K \tag{52}$$

$$y_{im} \in \{0,1\} \qquad z_m, dur[i,k] \geq 0$$

Before discussing the model in detail, several general concepts need to be clarified. In ILOG OPL-Studio, that was used to combine MILP and CP, the two main components are activities and resources. Resources are, for instance, machines and activities are processes that require the resources, such as processing a job in a stage. In this CP problem, *Stage* is an activity that has a given duration. Its starting and finishing times, $Stage.start, Stage.end$ are variables that will be evaluated during the solution procedure. Furthermore, $tool[m]$ is a set of unary resources, i.e. resources that can process only one activity at a time. The other resource, s, is an alternative resource of *tool* specifying that any of the given resources can be selected to perform the task. CP uses these concepts when assigning the activities to the available resources in order to generate feasible solutions.

The objective function (42) is exactly as in the MILP model. Eqs. (43)-(44) state that the starting time of the first stage should be greater than the release date and the finishing time for the last stage of each job should be less than the due date. Constraint (45) requires one of the alternative resources to process each stage. Due to the fact that we have a multistage problem with dissimilar parallel equipment, the duration of an activity is defined as a variable, the value of which is specified in Eq. (46) depending on the assignment. Constraint (47) eliminates the use of equipment from other stages and in Eq. (48) forbidden job-machine combinations are enforced. Note the use of the global constraint activityHasSelectedResource. Forbidden paths are enforced in constraint (49) and the order of stages is specified in constraint (50). Constraint (51) is needed only to link the objective function variables to the CP-model. The domain of the duration variable, $dur[i,k]$, is reduced in the last constraint (52).

The CP model is more compact than the MILP, but may not necessarily perform well with a given objective function, which only considers assignment costs. In general, CP is better suited for solving feasibility problems.

5.3. Results for MILP and CP models

Ten multistage scheduling problems, the data of which are presented in Appendix B, were solved using both the MILP and the CP strategies. The problem sizes and the results are presented in Table 3. It can be seen that although both representations can fully capture the problems, they both suffer from combinatorial explosion.

Table 3. Multistage scheduling results using MILP and CP models

Problem,	Obj	MILP			CP		
data	value	vars (0/1)	constr	CPU-s	vars	constr	CPU-s
1,1	39	60 (48)	96	0.13	84	140	0.02
1,2	112	60 (48)	96	0.03	84	140	0.02
2,1	53	112 (96)	190	4.43	124	210	0.13
2,2	188	112 (96)	190	0.78	124	210	0.32
3,1	56	198 (176)	463	446.60	182	409	4.23
3,2	1113	198 (176)	447	0.35	182	393	1374.97
4,1	149	396 (360)	718	27.30	306	772	447.17
4,2	946	396 (360)	718	1.43	306	772	359.01
5,1	111	572 (528)	1330	4041.66	392	1226	292.98
5,2	704	572 (528)	1330	2767.83	392	1226	712.61

Table 3 shows the results of five problems with two different data sets. The significance of the chosen data can be seen very clearly, especially in problems 3,1 and 3,2 where the performance varies by several magnitudes. It is interesting to note that MILP and CP show almost opposite behaviors. These problems have also different forbidden paths (none in 3,2), which explains the dissimilarity in the number of constraints. The results show that MILP over-performs CP in only a few of the problems, most likely because of its capability of using lower and upper bounds to eliminate large sets of feasible solutions. To summarize, the two methods are very different in performance, as can be clearly observed in the four largest problems.

5.4. Strategy I: Hybrid MILP-CP model

In this section we will investigate if Strategy I for the single-stage hybrid approach can be further developed to handle multistage scheduling. This involves several complications, for instance, the dimension of the problem is multiplied by each stage. Furthermore, in the multistage case we cannot solve the sequencing problem separately for each machine. Instead, all equipment need to be considered simultaneously in order to be able to coordinate between the stages and ensure that the obtained schedule is feasible. Thus, Strategy I will alternate between the solution of one MILP assignment problem and the solution of one CP sequencing problem. It is also important to observe that rather than determining whether a CP subproblem is feasible or infeasible, it is convenient for the generation of cuts to relax the due dates in the CP-subproblem by introducing slacks, which are then minimized to zero in the objective function. The slightly modified strategy is shown in Fig. 6.

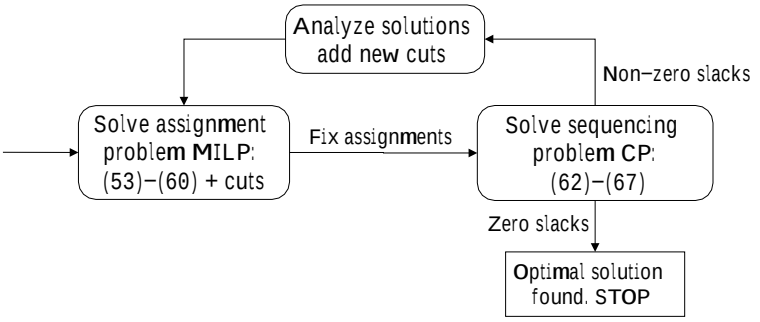

Figure 6. The multistage hybrid strategy

Here we actually need to perform optimization in the CP-step, instead of doing a simple feasibility check. This can be expected to significantly slow down the performance. The hybrid MILP-CP mathematical model for Strategy I is then as follows,

$$min \quad \sum_{i\in I}\sum_{m\in M} C'_{im}\cdot y_{im} + \sum_{m\in M} C''_{m}\cdot z_m \tag{53}$$

subject to

$$\sum_{m\in M(k)} y_{im} = 1 \qquad \forall i\in I,\ \forall k\in K \tag{54}$$

$$y_{im} \le z_m \qquad \forall i\in I,\ \forall m\in M \tag{55}$$

$$c_{ik} \le T_i^d \qquad \forall i\in I, k=|K| \tag{56}$$

$$c_{ik} \ge \sum_{m\in M(k)} y_{im}\cdot (T_i^r + T_m^s + T_{im}^p) \qquad \forall i\in I,\ \forall k\in K \tag{57}$$

$$y_{im} = 0 \qquad \forall i\in I,\ \forall m\in M, (i,m)\in \hat{B} \tag{58}$$

$$y_{im_1} + y_{im_2} \le 1 \qquad \forall i\in I,\ \forall (m_1,m_2)\in \hat{M} \tag{59}$$

$$\sum_{i\in I} y_{im}\cdot (T_{im}^p + T_m^s) \le \max_{i\in I}\left\{ T_i^d - \sum_{k'\in K, k'>k} \min_{m'\in M(k'),(i,m')\notin\hat{B}} (T_{im'}^p + T_{m'}^s)\right\} -$$

$$\min_{i\in I}\left\{ T_i^r + \sum_{k'\in K, k'<k} \min_{m'\in M(k'),(i,m')\notin\hat{B}} (T_{im'}^p + T_{m'}^s)\right\} \qquad \forall k\in K,\ \forall m\in M(k) \tag{60}$$

$$y_{im}\in\{0,1\} \qquad z_m, c_{ik}\ge 0$$

$$\text{if } (y_{im}=1) \text{ then } (z_{ik}=m) \quad \forall i\in I,\ \forall k\in K,\ \forall m\in M(k) \tag{61}$$

$$min \quad \sum_{i\in I} (C'_i\cdot s_i + C''_i\cdot l_i) \tag{62}$$

subject to

$$Stage[i,1].\text{start} \ge T_i^r \qquad \forall i\in I \tag{63}$$

$$Stage[i,|K|].\text{end} \leq T_i^d + s_i \qquad \forall i \in I \tag{64}$$

$$s_i > 0 \;\Leftrightarrow\; l_i = 1 \qquad \forall i \in I \tag{65}$$

$$Stage[i,k] \text{ requires } tool[m] \qquad \forall i \in I,\ \forall k \in K,\ \forall m \in M(k), z_{ik} = m \tag{66}$$

$$Stage[i,k] \text{ precedes } Stage[i,\text{next}(k)] \qquad \forall i \in I,\ \forall k \in K \mid k < |K| \tag{67}$$

$$s_i, l_i \geq 0$$

The MILP assignment part of the problem is given by Eqs. (53)-(60), that are exactly as in the original MILP model. Only the complicating sequencing variables and constraints have been removed. The link between the assignment and the corresponding CP sequencing problem is created by Eq. (61). The sequencing problem in Eqs. (62)-(67) is formulated for a given assignment. The objective function (62) minimizes a weighted sum of the total violation of the due dates and the number of late orders through a slack variable that relaxes the due dates, as defined in Eq. (64). Here, we use the coefficients $C_i' = 1$ and $C_i'' = 1000$. Equation (65) activates the flag ($l_i = 1$) if product i is late. The fixed assignment is transferred to the CP-problem in constraint (66). Constraints (63) and (67) are identical to the original CP model. This formulation is used to first find an assignment and then try to get a valid schedule for it. After each iteration, cuts need to be added to eliminate invalid assignments.

5.5. Integer Cuts

The integer cuts for the MILP assignment subproblem are maybe the most critical elements in the strategy. The simplest alternative is to eliminate all assignments with only one cut. The general form of the cut is given in Eq. (68) and the cut is determined by the coefficients a_{im}. The result of an assignment problem is illustrated by Fig. 7a, where 6 jobs are assigned to 2 stages, each of which have two parallel equipment.

$$\sum_{i \in I} \sum_{m \in M} a_{im} \cdot y_{im} \leq \sum_{i \in I} \sum_{m \in M} a_{im} - 1 \tag{68}$$

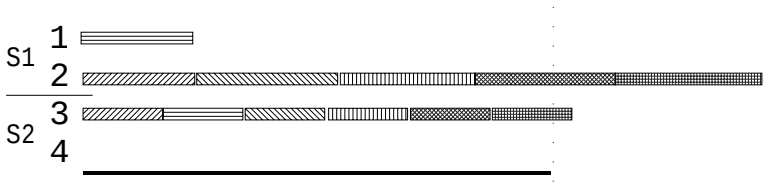

Figure 7a. Optimal assignment

Having a fixed assignment, we need to find a sequence that meets the due dates. In the multistage case, a feasibility problem is not solved, but an optimization problem minimizing the number of late jobs and the slack variables. The result of the sequencing problem is shown in Fig. 7b where it can be observed that a working schedule can not be generated using the given assignment.

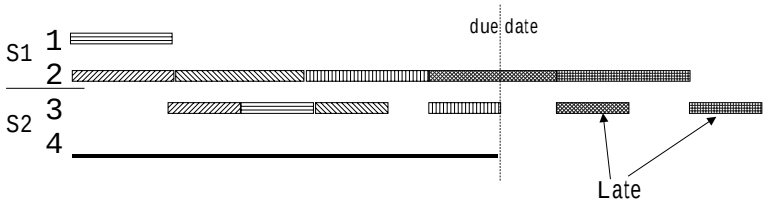

Figure 7b. Best sequence for the fixed assignment

The assignment is therefore invalid and must be excluded from future iterations. If the products are denoted by A,B,C,D,E,F starting from the top-left, the cut that excludes this exact assignment, CUT 1, can be written as,

$$y_{A1} + y_{B2} + y_{C2} + y_{D2} + y_{E2} + y_{F2} + y_{A3} + y_{B3} + y_{C3} + y_{D3} + y_{E3} + y_{F3} \leq 11 \tag{69}$$

This cut excludes only the given assignment from the next iteration and does not exclude any supersets or more general cases. Consequently, the cut is weak and can be expected to result in a large number of iterations for solving the MILP and CP subproblems. This is shown in the example runs in Table 5.

A stronger cut is necessary for obtaining better performance. In the following, CUT 2 that focuses on the late jobs is suggested. This cut does not include all products or machines and therefore it will exclude a larger number of possible solutions. The cut eliminates the bottlenecking assignments by tracing for each late job a zero-wait path from the last stage to the first stage. This strategy produces one cut per late job and the principle is given below.

For each late job:

select the late job in its last stage ($k = |K|$), add it to the cut ($a_{jm} = 1, m \in M(k)$) and find every bottleneck path. There are two possibilities.

1. if selected job follows immediately another job in the same stage (machine), select the other job and add it to the cut ($a_{j-1,m} = 1$)
2. else, move to previous stage ($k = k - 1$) and add the assignment ($a_{jm} = 1, m \in M(k)$) to the cut

There must be at least one bottleneck path for each late job from the last stage to zero-time (or release date) because of the minimization of slack variables. Applying this principle to the earlier example would result in two cuts for CUT 2:

$$y_{B2} + y_{C2} + y_{D2} + y_{E2} + y_{E3} \leq 4 \tag{70}$$

$$y_{B2} + y_{C2} + y_{D2} + y_{E2} + y_{F2} + y_{F3} \leq 5 \tag{71}$$

These cuts are more efficient, since they cut out a large number of supersets. Consequently, the convergence should also be faster. A drawback is that these cuts are strictly speaking heuristic since in few cases they may cut off the true optimal solution. However, we would like to emphasize that we have found CUT 2 to be valid in most of the problems we have solved (e.g. see Table 5). An alternative cut, CUT 3, can be obtained if the second step is modified to,

2. else, add all earlier assignments on the machine to the cut, move to previous stage ($k = k - 1$) and add the assignment ($a_{jm} = 1, m \in M(k)$) to the cut

It should be noted that CUT 3 is still heuristic, although it is valid for most of the cases we tested. In the example case, this will result in two weaker cuts,

$$y_{B2} + y_{C2} + y_{D2} + y_{E2} + y_{A3} + y_{B3} + y_{C3} + y_{D3} + y_{E3} \leq 8 \tag{72}$$

$$y_{B2} + y_{C2} + y_{D2} + y_{E2} + y_{F2} + y_{A3} + y_{B3} + y_{C3} + y_{D3} + y_{E3} + y_{F3} \leq 10 \qquad (73)$$

In general, the fewer products a cut involves, the stronger it can be considered. These three different cuts are referred to as CUT 1, CUT 2 and CUT 3. The three example cases are presented in Figures 8a, 8b and 8c. In the two latter figures, the cuts are illustrated with gray color and hatching.

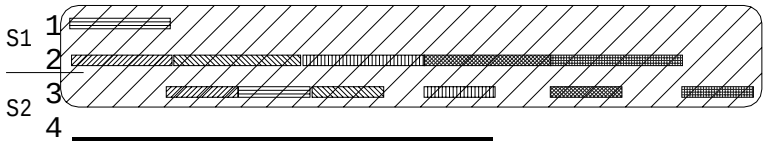

Figure 8a. CUT 1

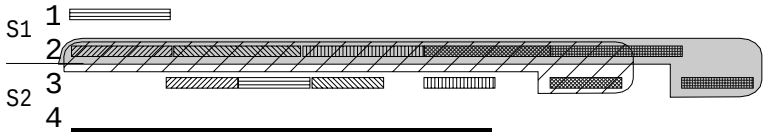

Figure 8b. CUT 2

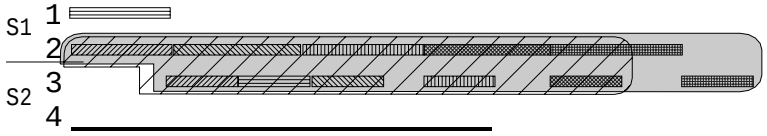

Figure 8c. CUT 3

It should be emphasized that only CUT 1 can theoretically guarantee the optimal solution. While we have found CUT 2 and CUT 3 to be valid in most problems, we show in Appendix C two specific example problems where the either or both of these two cuts eliminate the optimal solution. The results of these problems are given in Table 4.

Table 4. Non-optimal example runs (CPU-s/it/cuts)

Problem	Optimal	MILP	CUT 1	CUT 2	CUT 3
Example 1	3761	3761[1] (60000)	3704[2] (1000)	3781 (27.52/42/48)	3761 (259.42/158/273)
Example 2	427	427 (55.08)	427[2] (1000)	434 (51.81/53/71)	431 (655.44/284/393)

[1)] Optimal solution found, but not verified at the CPU-time limit

[2)] Non-confirmed solution at the CPU-time limit (lower bound)

Table 4 presents the solutions, where it can be seen that the objectives obtained with CUT 2 and CUT 3 are higher. For the different strategies we report the solution and CPU-time.

5.6. Results for Strategy I

The example problems in Table 3 were resolved using hybrid Strategy I and applying all three cut strategies discussed above. In Table 5, we report the CPU-times (MILP and CP), the number of iterations and cuts needed by each of the strategies. The solution process was stopped after a given time limit, 1000 CPU-s. Note that in this case solution is infeasible because it does not meet the due dates. These cases are marked with lines (–). It should be noted that in all cases the CUT 2 strategy obtained the rigorous solutions.

Table 5. Multistage scheduling results with hybrid MILP and CP Strategy I

Problem,	CUT 1*		CUT 2		CUT 3*	
data	it/cuts	CPU (MILP/CP)	it/cuts	CPU (MILP/CP)	it/cuts	CPU (MILP/CP)
1,1	5/4	0.11 (0.06/0.05)	5/4	0.06 (0.04/0.02)	5/4	0.11 (0.06/0.05)
1,2	3/2	0.05 (0.03/0.02)	3/2	0.03 (0.02/0.01)	3/2	0.04 (0.03/0.01)
2,1	259/258	106.38 (99.22/7.16)	16/19	0.74 (0.33/0.41)	45/73	3.24 (2.09/1.15)
2,2	16/15	0.43 (0.19/0.24)	4/3	0.10 (0.05/0.05)	5/4	0.13 (0.07/0.06)
3,1	–	–	29/45	8.69 (4.20/4.49)	87/179	59.54 (45.47/14.07)
3,2	74/73	11.25 (7.82/3.43)	5/5	0.38 (0.07/0.31)	7/7	0.40 (0.12/0.28)
4,1	–	–	17/22	28.76 (0.84/27.92)	98/182	236.95 (39.10/197.85)
4,2	–	–	20/26	10.95 (0.90/10.05)	33/47	13.72 (2.05/11.67)
5,1	–	–	43/56	318.41 (21.99/296.42)	–	–
5,2	2/1	13.72 (0.06/13.64)	2/2	13.73 (0.08/13.65)	2/2	13.59 (0.06/13.53)

*) Entries with '–' were stopped after 1000 CPU-s

The results show that CUT 1 is generally not very effective for this type of scheduling problems, even though it is the only one that can guarantee an optimal solution to the multistage problem. The weak cuts result in very large a number of iterations for the fairly small problem 2,1. Compared to the pure MILP and CP strategies of Table 3, CUT 2 performs very well and the number of cuts is low. Only example problem 4,1 is somewhat slower than the original MILP solution, and example 5,1 takes slightly longer than the original CP solution. It can also be noted that the CP-subproblems are large and often take almost the entire CPU-time. The cuts are therefore extremely important to the performance. CUT 3 is not able to perform as well as CUT 2, mostly because of the increased number of iterations. From these example problems, we can see that the rigorous CUT 1 is not applicable for larger multistage scheduling problems. However, CUT 2 emerged as the best choice since it is able to reduce the computational time by one order of magnitude.

5.7. Results for Strategy II

Because of the fact that optimization is required in the sequencing part (minimization of slacks), the MILP subproblems for Strategy II could be expected to be well suited for the multistage problem. The MILP formulation for the multistage sequencing problem can be obtained with minor changes. It is given in Eqs. (74)-(80).

$$min \quad \sum_{i \in I} (C'_i \cdot s_i + C''_i \cdot l_i) \tag{74}$$

subject to

$$c_{ik} \leq T^d_i + s_i \qquad \forall i \in I, k = |K| \tag{75}$$

$$c_{ik} \geq T^r_i + T^s_m + T^p_{im} \qquad \forall i \in I,\ \forall k \in K,\ \forall m \in M(k), y_{im} = 1 \tag{76}$$

$$s_i \leq U \cdot l_i \qquad \forall i \in I \tag{77}$$

$$c_{ik} \leq c_{i,k+1} - T^p_{im} - T^s_m \qquad \forall i \in I, \forall k \in K, k < |K|,\ \forall m \in M(k+1), y_{im} = 1 \tag{78}$$

$$c_{i'k} \geq c_{ik} + T^p_{i'm} + T^s_m - \mathrm{U} \cdot (1 - x_{ii'k})$$
$$\forall i, i' \in I, i < i',\ \forall k \in K,\ \forall m \in M(k), y_{im} = y_{i'm} = 1 \tag{79}$$

$$c_{ik} \geq c_{i'k} + T^p_{im} + T^s_m - \mathrm{U} \cdot x_{ii'k}$$

$$\forall i, i' \in I, i < i',\ \forall k \in K,\ \forall m \in M(k), y_{im} = y_{i'm} = 1 \tag{80}$$

$$x_{ii'k}, l_i \in \{0,1\} \qquad c_{ik}, s_i > 0$$

In the following, we will revisit the examples and only use the heuristic, CUT 2, to see how Strategy II performs. The results are given in Table 6.

Table 6. Multistage scheduling with Strategy II using CUT 2

Problem,	CUT 2	
data	it/cuts	CPU (MILP/MILP)
1,1	5/4	0.19 (0.04/0.15)
1,2	3/2	0.11 (0.03/0.08)
2,1	16/19	2.82 (0.31/2.51)
2,2	4/3	0.70 (0.05/0.65)
3,1	40/61	57.76 (11.33/46.43)
3,2	5/4	0.72 (0.06/0.66)
4,1	22/32	435.46 (1.23/434.23)
4,2	16/21	12.97 (0.53/12.44)
5,1	– *	–
5,2	2/2	540.30 (0.19/540.11)

*) Could not be solved in 1000 CPU-s

The optimization results in Table 6 show that Strategy II performs slightly worse than Strategy I (see Table 5) because the MILP sequencing subproblems take more time than the CP subproblems. Problem 5,1 could not even be solved within the given time limit (1000 CPU-s) and the reason for this is most likely the poor relaxations combined with large number of discrete variables. It seems that CP with its specific scheduling constructs is more efficient in solving the sequencing part of the hybrid strategy than the more general MILP, also in the case where the feasibility check involves optimization of some slack variables.

6. CONCLUSIONS

The main objective of this paper has been to classify and consider methods for using hybrid models for solving scheduling problems that involve cost minimization with due dates, and sequence independent setup times. It was shown that decomposition schemes can be developed in which the MILP effectively solves very well the assignment problem, and either CP or MILP are applicable to the sequencing part. The single-stage approach in Jain and Grossmann (2001), that uses CP for sequencing was revisited, and the Strategy II, in which MILP is used for both subproblems, was proposed for solving the single-stage problem. The algorithm proved to be robust with different data sets. It could also be concluded that replacing the CP-part of the single-stage problem with MILP actually dropped the performance by almost an order of magnitude, but was still significantly superior in comparison to using only MILP or only CP models. Motivated by the encouraging results of a single-stage case, Strategies I and II were applied to the more complex multistage scheduling problem. Two cuts (CUT 2, CUT 3) were proposed which were found to greatly reduce the computational times, with CUT 2 being the more effective of the two. Although these cuts are not rigorous, they were found to yield the correct solution in most of the test problems. Finding a rigorous and strong cut for the hybrid strategy, improving the performance of the sequencing problem or disaggregating the large-size

problems in some other way are open questions to be explored in future work for the multistage case.

ACKNOWLEDGMENTS

The Center for Advanced Process Decision-making (CAPD) at Carnegie Mellon University is gratefully acknowledged for financial support for this work. We would also like to thank Professor José M. Pinto for interesting suggestions regarding this work.

REFERENCES

Balas E., Ceria S. and Cornuejols G. (1996). Mixed 0–1 Programming by Lift-and-Project in a Branch-and-Cut Framework. *Management Science*, **42**, pp. 1229–1246.

Carlier J. and Pinson E. (1989). An Algorithm for Solving the Job-Shop Problem. *Management Science*, **35** (2), pp. 164–176

Dash Associates (1999). XPRESS-MP, User Guide.

Dincbas M., Van Hentenryck P., Simonis H., Aggoun A., Graf T. and Berthier F. (1988). The Constraint Logic Programming Language CHIP. *FGCS 1988, Proceedings of the International Conference on Fifth Generation Computer Systems 1988, Tokyo*, pp. 693–702.

Harjunkoski I. and Grossmann I.E. (2001). Combined MILP-Constraint Programming Approach for the Optimal Scheduling of Multistage Batch Processes, *Computer-aided Chemical Engineering*, **9**, pp. 877–882

Harjunkoski I., Jain V. and Grossmann I.E. (2000). Hybrid Mixed-Integer/Constrained Logic Programming Strategies for Solving Scheduling and Combinatorial Optimization Problems, *Computers chem. Engng*, **24**, pp. 337–343

Hentenryck P.V. (1989). Constraint Satisfaction in Logic Programming. *MIT Press, Cambridge, MA*

Hooker J. (2000) Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction. *John Wiley and Sons, Inc., New York*

IBM (1992). IBM Optimization Subroutine Library Guide and Reference. *IBM Systems Journal SC23-0519*, **31** (1).

Ierapetritou M. and Floudas C.A. (1998). Effective continuous-time formulation for short-term scheduling. *Industrial and Engineering Chemistry Research*, **37**, pp. 4341–4359.

ILOG CPLEX 7.0. User's Manual (2000). *ILOG Inc.*

ILOG OPL Studio 3.5. The Optimization Language (2001). *ILOG Inc.*

ILOG Solver 4.4. User's Manual (1999). *ILOG Inc.*

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

Johnson E.L., Nemhauser G.L. Savelsbergh M.W.P. (2000). Progress in Linear Programming Based Branch and Bound Algorithm: Exposition. *INFORMS Journal of Computing*, **12** (1), pp. 2–23

Marriott K. and Stuckey P.J. (1999). Introduction to Constraint Logic Programming. *MIT press., Cambridge*

Méndez C. A., Henning G. P. and Cerdá J. (2000). An MILP Continuous-Time Approach to Short-Term Scheduling of Resource-Constrained Multistage Flowshop Batch Facilities. *Computers chem. Engng*, **25**, pp. 701–711.

Nemhauser G.L. and Wolsey L.A. (1988). Integer and Combinatorial Optimization. *John Wiley and Sons, Inc., New York*

Ottosson G., Thorsteinsson E.S. and Hooker J.N. (2001). Mixed Global Constraints and Inference in Hybrid CLP-IP Solvers. *accepted for publication in Annals of Mathematics and Artificial Intelligence*

Pekny J.F. and Reklaitis G.V. (1998). Towards the Convergence of Theory and Practice: A Technology Guide for Scheduling/Planning Methodology. *AIChE Symposium Series*, **94** (320), pp. 91–111.

Pinto J. and Grossmann I.E. (1995). A Continuous Time Mixed Integer Linear Programming Model for Short Term Scheduling of Multistage Batch Plants. *I&EC Research*, **34**, pp. 3037–3051.

Pinto J. and Grossmann I.E. (1998). Assignment and Sequencing Models for the Scheduling of Chemical Processes. *Annals of Operations Research*, **81**, pp. 433–466.

Raman R. and Grossmann I.E. (1994). Modeling and Computational Techniques for Logic Based Integer Programming. *Computers chem. Engng*, **18**, pp. 563–578.

Shah N. (1998). Single- and Multisite Planning and Scheduling: Current Status and Future Challenges, *AIChE Symposium Series*, **94** (320), pp 75–90.

Wallace M., Novello S., and Schimpf J. (1997). ECLiPSe: A Platform for Constraint Logic Programming. *ICL Systems Journal*, **12** (1), pp. 159–200.

Wang L. and Zheng D-Z. (2001). An Effective Hybrid Optimization Strategy for Job-Shop Scheduling Problems, *Computers & Operations Research*, **28**, pp. 585–596.

APPENDIX A

The data used for the single stage scheduling problems is given in the tables below.

Table 7. Single stage problem 1 (2 machines, 3 jobs)

	Data set 1				Data set 2			
	dates		duration/cost		dates		duration/cost	
Job	release	due	M1	M2	release	due	M1	M2
1	20	169	103/10	143/6	20	160	57/10	77/6
2	30	169	63/8	83/5	30	130	37/8	47/5
3	40	219	113/12	163/7	40	210	57/12	77/7

Table 8. Single stage problem 2 (3 machines, 7 jobs)

	Data set 1					Data set 2				
	dates		duration/cost			dates		duration/cost		
Job	release	due	M1	M2	M3	release	due	M1	M2	M3
1	20	169	103/10	143/6	123/8	20	160	57/10	77/6	67/8
2	30	139	63/8	83/5	73/6	30	130	37/8	47/5	37/6
3	40	219	113/12	163/7	133/10	40	210	27/12	47/7	37/10
4	50	289	63/10	123/6	83/8	50	280	37/10	67/6	47/8
5	100	249	103/8	163/5	123/7	100	240	27/8	47/5	37/7
6	10	289	73/12	123/7	103/10	10	280	17/12	37/7	27/10
7	20	239	103/12	83/7	103/10	20	230	17/12	27/7	17/10

Table 9. Single stage problem 3 (3 machines, 12 jobs)

	Data set 1					Data set 2				
	dates		duration/cost			dates		duration/cost		
Job	release	due	M1	M2	M3	release	due	M1	M2	M3
1	20	360	103/10	143/6	123/8	20	360	57/10	77/6	67/8
2	30	330	63/8	83/5	73/6	30	330	37/8	47/5	37/6
3	40	310	113/12	163/7	133/10	40	310	27/12	47/7	37/10
4	50	380	63/10	123/6	83/8	50	380	37/10	67/6	47/8
5	100	340	103/8	163/5	123/7	100	340	27/8	47/5	37/7
6	10	380	73/12	123/7	103/10	10	380	17/12	37/7	27/10
7	20	330	103/12	133/10	103/11	20	330	17/12	27/10	17/11
8	40	250	43/9	103/5	83/7	40	250	27/9	57/5	47/7
9	100	380	23/10	43/6	33/8	100	380	47/10	67/6	67/8
10	10	390	73/8	143/5	113/7	10	390	37/8	57/5	27/7
11	50	300	83/15	163/9	123/12	50	300	27/15	37/9	27/12
12	20	200	33/13	63/7	53/10	20	200	27/13	67/7	47/10

Table 10. Single stage problem 4 (5 machines, 15 jobs)

	Data set 1							Data set 2						
	dates		duration/cost					dates		duration/cost				
Job	release	due	M1	M2	M3	M4	M5	release	due	M1	M2	M3	M4	M5
1	20	330	103/10	143/6	123/8	113/9	133/9	20	330	57/10	77/6	67/8	57/9	67/9
2	30	340	63/8	83/5	73/6	63/7	73/7	30	340	37/8	47/5	37/6	37/7	47/7
3	40	310	113/12	163/7	133/10	113/11	123/10	40	310	27/12	47/7	37/10	27/11	37/10
4	50	330	63/10	123/6	83/8	73/9	83/8	50	330	37/10	67/6	47/8	37/9	47/8
5	100	340	103/8	163/5	123/6	123/7	133/7	100	340	27/8	47/5	37/6	27/7	27/7
6	10	340	73/12	123/7	103/10	83/11	93/10	10	340	17/12	37/7	27/10	27/11	27/10
7	20	330	103/12	133/10	103/11	113/12	123/11	20	330	17/12	27/10	17/11	17/12	17/11
8	40	250	43/9	103/5	83/7	53/9	63/8	40	250	27/9	57/5	47/7	37/9	37/8
9	100	380	23/10	43/6	33/8	23/9	33/8	100	380	47/10	67/6	67/8	57/9	57/8
10	10	370	73/8	143/5	113/6	83/7	103/6	10	370	27/8	57/5	37/6	27/7	37/6
11	50	300	83/15	163/9	123/12	103/14	113/13	50	300	27/15	37/9	27/12	27/14	27/13
12	20	200	33/13	63/7	53/10	43/12	53/11	20	200	27/13	67/7	47/10	37/12	37/11
13	40	320	43/9	103/5	73/6	53/8	63/7	40	320	17/9	37/5	37/6	27/8	27/7
14	60	200	23/10	43/6	43/8	33/10	33/9	60	200	27/10	57/6	57/8	27/10	37/9
15	20	250	73/8	143/5	133/6	103/7	113/7	20	250	47/8	77/5	67/6	47/7	57/7

Table 11. Single stage problem 5 (5 machines, 20 jobs)

	Data set 1							Data set 2						
	dates		duration/cost					dates		duration/cost				
Job	release	due	M1	M2	M3	M4	M5	release	due	M1	M2	M3	M4	M5
1	20	330	103/10	143/6	123/8	113/9	133/9	20	330	57/10	77/6	67/8	57/9	67/9
2	30	340	63/8	83/5	73/6	63/7	73/7	30	340	37/8	47/5	37/6	37/7	47/7
3	40	310	113/12	163/7	133/10	113/11	123/10	40	310	27/12	47/7	37/10	27/11	37/10
4	50	330	63/10	123/6	83/8	73/9	83/8	50	330	37/10	67/6	47/8	37/9	47/8
5	100	340	103/8	163/5	123/6	123/7	133/7	100	340	27/8	47/5	37/6	27/7	27/7
6	10	340	73/12	123/7	103/10	83/11	93/10	10	340	17/12	37/7	27/10	27/11	27/10
7	20	330	103/12	133/10	103/11	113/12	123/11	20	330	17/12	27/10	17/11	17/12	17/11
8	40	250	43/9	103/5	83/7	53/9	63/8	40	250	27/9	57/5	47/7	37/9	37/8
9	100	380	23/10	43/6	33/8	23/9	33/8	100	380	47/10	67/6	67/8	57/9	57/8
10	10	370	73/8	143/5	113/6	83/7	103/6	10	370	27/8	57/5	37/6	27/7	37/6
11	50	300	83/15	163/9	123/12	103/14	113/13	50	300	27/15	37/9	27/12	27/14	27/13
12	20	200	33/13	63/7	53/10	43/12	53/11	20	200	27/13	67/7	47/10	37/12	37/11
13	40	320	43/9	103/5	73/6	53/8	63/7	40	320	17/9	37/5	37/6	27/8	27/7
14	60	200	23/10	43/6	43/8	33/10	33/9	60	200	27/10	57/6	57/8	27/10	37/9
15	20	250	73/8	143/5	133/6	103/7	113/7	20	250	47/8	77/5	67/6	47/7	57/7
16	20	340	33/9	83/5	73/7	53/9	63/8	20	340	27/9	47/5	37/7	27/9	37/8
17	30	370	63/10	123/6	103/8	73/9	83/8	30	370	37/10	67/6	47/8	37/9	47/8
18	70	380	23/8	83/5	63/6	133/7	43/6	70	380	27/8	47/5	37/6	27/7	27/6
19	60	320	43/15	73/9	63/12	53/14	53/13	60	320	17/15	37/9	27/12	27/14	27/13
20	0	300	53/13	73/7	73/10	63/12	63/11	0	300	17/13	27/7	17/10	17/12	17/11

APPENDIX B

The data for the multistage scheduling problems is given in the tables below. The data given above the table refers to both data sets. Forbidden assignments are marked with the processing time 'F'.

Table 12. Multistage problem 1 (4 machines, 2 stages, 4 jobs)

Stage 1 = [M1,M2], Stage 2 = [M3,M4]

	Data set 1						Data set 2					
	dates		duration/cost				dates		duration/cost			
Job	release	due	M1	M2	M3	M4	release	due	M1	M2	M3	M4
1	20	800	171/1	171/1	110/1	96/1	20	800	115/14	120/13	110/8	196/2
2	50	600	111/2	100/2	86/1	96/1	50	800	120/12	110/12	110/11	196/11
3	0	800	151/1	151/1	96/1	96/1	10	800	120/12	110/12	110/11	196/11
4	0	1000	161/1	161/3	90/1	96/1	100	800	160/10	160/25	90/32	222/10
	Setup[m] = [80,80,40,40]						Setup[m] = [20,25,43,20]					

Table 13. Multistage problem 2 (4 machines, 2 stages, 6 jobs)

Stage 1 = [M1,M2], Stage 2 = [M3,M4]

	Data set 1						Data set 2					
	dates		duration/cost				dates		duration/cost			
Job	release	due	M1	M2	M3	M4	release	due	M1	M2	M3	M4
1	0	800	171/1	171/1	96/1	96/1	100	800	151/10	171/10	86/10	96/10
2	0	600	111/2	111/2	96/1	96/1	200	600	101/20	111/20	86/13	96/10
3	0	800	151/1	151/1	96/1	96/1	10	800	131/18	151/18	86/13	96/10
4	0	1000	161/2	161/1	96/1	96/1	0	600	121/28	161/14	76/13	96/10
5	0	800	161/1	161/1	96/1	96/1	50	800	141/16	161/16	86/13	96/10
6	0	800	111/1	111/3	96/1	96/1	0	800	91/10	111/30	80/12	96/10
	Setup[m] = [80,80,40,40]						Setup[m] = [20,40,20,20]					

Table 14. Multistage problem 3 (6 machines, 2 stages, 8 jobs)

Stage 1 = [M1,M2,M3], Stage 2 = [M4,M5,M6]

	Data set 1								Data set 2							
	dates		duration/cost						dates		duration/cost					
Job	release	due	M1	M2	M3	M4	M5	M6	release	due	M1	M2	M3	M4	M5	M6
1	0	800	F	171/1	171/1	95/1	96/1	96/1	200	800	F	171/85	171/90	95/100	96/100	96/110
2	0	600	101/2	111/2	111/1	95/1	96/1	96/1	100	600	101/50	111/52	111/41	95/91	96/91	96/91
3	0	800	141/1	151/1	151/1	95/1	96/1	96/1	50	800	141/100	151/110	151/90	95/50	96/50	96/55
4	0	1000	160/2	161/1	161/1	95/1	96/1	96/1	0	1000	160/20	161/10	161/10	95/61	96/61	96/61
5	0	800	160/1	161/1	161/1	95/1	96/1	96/1	0	800	160/80	161/80	161/90	95/110	96/120	96/130
6	0	800	160/1	161/1	161/1	95/1	96/1	96/1	0	800	160/70	161/70	161/70	95/40	96/40	96/30
7	0	1500	160/1	161/1	161/1	95/1	96/1	96/1	300	1500	160/100	161/100	161/100	95/90	96/90	96/100
8	0	1200	111/1	111/3	111/1	95/1	96/1	96/1	200	1200	111/10	111/30	111/40	95/45	96/45	96/45
	Setup[m] = [80,80,75,40,40,40] Forbidden paths = [M1→M6, M2→M5]								Setup[m] = [80,80,75,40,40,40]							

Table 15. Multistage problem 4 (6 machines, 3 stages, 10 jobs)

Stage 1 = [M1,M2], Stage 2 = [M3,M4], Stage 3 = [M5,M6]

	Data set 1								Data set 2							
	dates		duration/cost						dates		duration/cost					
Job	release	due	M1	M2	M3	M4	M5	M6	release	due	M1	M2	M3	M4	M5	M6
1	0	800	181/4	F	120/4	121/4	95/2	96/2	100	800	181/46	F	120/24	121/24	95/22	95/32
2	0	600	101/2	111/2	120/3	121/3	95/2	96/2	50	600	101/42	111/52	120/23	121/23	95/33	95/43
3	0	800	141/3	151/4	110/2	111/2	95/2	F	0	700	141/39	151/34	110/14	111/12	93/13	F
4	0	1000	160/5	161/5	120/3	121/4	95/2	96/2	300	1100	160/15	161/15	120/13	121/14	92/13	95/12
5	0	2000	160/5	161/5	120/3	121/4	95/2	96/2	400	2000	160/25	161/32	120/32	121/42	91/13	95/15
6	0	1800	160/6	161/5	120/3	121/3	95/2	96/2	200	1800	140/61	161/56	120/34	121/30	95/20	95/28
7	0	900	160/5	161/5	120/4	121/4	95/2	96/2	100	800	160/48	111/57	120/41	121/40	94/26	95/29
8	0	1200	160/6	161/5	120/3	121/4	95/2	96/2	200	1200	140/64	111/51	120/37	121/41	95/22	95/22
9	0	100	160/4	161/4	120/3	121/3	95/2	96/2	400	1600	140/36	101/45	120/39	121/40	95/16	95/15
10	0	1200	111/2	111/3	120/2	121/2	95/2	96/1	0	1200	111/22	111/33	120/27	121/39	95/25	95/38
	Setup[m] = [80,80,25,25,40,40]								Setup[m] = [80,80,25,25,40,40]							
	Forbidden paths = [M1⟶M3]								Forbidden paths = [M1⟶M3]							

Table 16. Multistage problem 5, data set 1 (8 machines, 3 stages, 12 jobs)

Stage 1 = [M1,M2,M3,M4], Stage 2 = [M5,M6], Stage 3 = [M7,M8]

	Data set 1									
	dates		duration/cost							
Job	release	due	M1	M2	M3	M4	M5	M6	M7	M8
1	100	800	151/1	131/1	F	120/1	100/2	95/1	90/2	80/3
2	0	600	101/1	111/1	111/1	120/1	100/2	95/1	90/2	80/3
3	140	800	141/1	151/2	162/2	120/1	100/2	95/1	F	80/3
4	200	1000	120/1	125/2	125/2	120/1	100/2	95/1	90/2	80/3
5	400	2000	120/1	125/2	125/2	120/1	100/2	95/1	90/2	80/3
6	0	800	120/1	125/2	125/3	120/1	100/2	95/1	90/2	80/3
7	150	900	120/1	120/2	120/3	120/1	100/2	95/1	90/2	80/3
8	200	1200	120/1	120/2	120/3	120/1	100/2	95/1	90/2	80/3
9	0	600	120/1	120/2	120/3	120/1	100/2	95/1	90/2	80/3
10	220	1200	120/1	120/2	120/3	120/1	100/2	95/1	90/2	80/3
11	600	2300	120/1	120/2	120/3	120/1	100/2	95/1	90/2	80/3
12	300	1800	111/1	111/2	112/3	120/1	100/2	95/1	90/2	80/3
	Setup[m] = [40,40,40,40,25,20,40,20]									
	Forbidden paths = [M1⟶M5,M2⟶M5,M5⟶M7,M6⟶M7]									

Table 17. Multistage problem 5, data set 2 (8 machines, 3 stages, 12 jobs)

Stage 1 = [M1,M2,M3,M4], Stage 2 = [M5,M6], Stage 3 = [M7,M8]

	Data set 1									
	dates		duration/cost							
Job	release	due	M1	M2	M3	M4	M5	M6	M7	M8
1	100	1800	151/10	131/10	F	110/11	110/20	85/21	90/22	90/23
2	100	1600	131/10	111/10	111/10	110/12	110/20	85/15	90/21	90/30
3	200	600	161/15	151/25	162/25	110/14	110/20	85/10	F	90/30
4	200	1200	135/10	125/22	125/22	110/12	110/20	85/13	90/22	90/34
5	400	1200	135/10	125/22	125/22	110/21	110/20	85/13	90/22	90/34
6	100	1400	135/10	125/22	125/23	110/21	110/20	85/13	90/22	90/34
7	150	700	130/10	120/22	120/23	120/22	100/20	85/13	90/22	90/34
8	200	1200	130/10	120/22	120/23	120/22	100/24	85/14	90/22	90/32
9	50	900	130/12	120/12	120/14	120/22	100/24	85/11	90/22	90/32
10	220	1300	130/12	120/12	120/14	120/22	100/24	85/12	90/20	90/32
11	600	2000	130/12	120/12	120/23	120/22	100/24	85/12	90/20	90/30
12	300	1400	121/12	111/12	112/23	120/10	100/22	85/12	90/20	90/30
	Setup[m] = [10,10,10,20,30,30,15,20]									
	Forbidden paths = [M1$\rightarrow$M5,M2$\rightarrow$M5,M5$\rightarrow$M7,M6$\rightarrow$M7]									

APPENDIX C

The two problems below explore the weakness of the heuristic cuts and demonstrates a case where an optimal solution could not be obtained by either CUT 2 and CUT 3.

Table 18. Example 1, where CUT 2 fails (6 machines, 3 stages, 10 jobs)

Stage 1 = [M1,M2], Stage 2 = [M3,M4], Stage 3 = [M5,M6]

	CUT 2 non-optimal							
	dates		duration/cost					
Job	release	due	M1	M2	M3	M4	M5	M6
1	0	700	151/151	F	120/120	121/121	135/135	96/96
2	0	900	101/202	211/422	120/120	121/121	125/125	96/96
3	0	1100	131/131	171/171	110/110	111/111	125/125	F
4	0	1000	140/280	191/191	120/120	121/121	115/230	96/96
5	0	1200	140/140	200/200	120/120	121/121	145/145	96/96
6	0	1300	130/130	151/151	120/120	121/121	135/135	96/96
7	0	1300	150/150	171/171	120/120	121/121	125/125	96/96
8	0	1000	110/110	141/141	120/120	121/121	115/115	96/96
9	0	1600	100/100	151/151	120/120	121/121	115/115	96/96
10	0	1200	91 /91	211/633	120/120	121/121	115/115	96/96
	Setup[m] = [60,80,25,25,40,40]							
	Forbidden paths = [M1$\longrightarrow$M3]							

Table 19. Example 2, where both CUT 2&3 fail (6 machines, 3 stages, 10 jobs)

Stage 1 = [M1,M2], Stage 2 = [M3,M4], Stage 3 = [M5,M6]

	CUT 2 & 3 non-optimal							
	dates		duration/cost					
Job	release	due	M1	M2	M3	M4	M5	M6
1	0	700	51/15	F	22/8	12/12	35/13	36/16
2	100	600	51/4	21/5	22/12	12/11	25/12	36/16
3	200	600	31/9	17/11	21/14	11/11	25/15	F
4	150	400	40/8	49/9	22/15	12/12	15/23	26/16
5	200	500	40/4	50/3	22/14	12/12	45/14	46/16
6	200	800	30/3	35/5	22/14	12/12	35/13	36/19
7	100	600	50/5	57/7	12/10	12/12	25/22	26/19
8	150	500	10/5	14/4	24/11	24/12	15/21	26/19
9	400	600	10/3	15/5	22/20	21/25	15/25	16/29
10	0	500	9 /9	11/6	22/20	21/21	15/25	16/29
	Setup[m] = [60,80,25,25,40,40]							
	Forbidden paths = [M1$\longrightarrow$M3]							