Algorithms for Hybrid MILP/CP Models for a Class of Optimization Problems

Vipul Jain • Ignacio E. Grossmann
Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213, USA
Vipul_Jain@i2.com • grossmann@cmu.edu

The goal of this paper is to develop models and methods that use complementary strengths of Mixed Integer Linear Programming (MILP) and Constraint Programming (CP) techniques to solve problems that are otherwise intractable if solved using either of the two methods. The class of problems considered in this paper have the characteristic that only a subset of the binary variables have non-zero objective function coefficients if modeled as an MILP. This class of problems is formulated as a hybrid MILP/CP model that involves some of the MILP constraints, a reduced set of the CP constraints, and equivalence relations between the MILP and the CP variables. An MILP/CP based decomposition method and an LP/CP-based branch-and-bound algorithm are proposed to solve these hybrid models. Both these algorithms rely on the same relaxed MILP and feasibility CP problems. An application example is considered in which the least-cost schedule has to be derived for processing a set of orders with release and due dates using a set of dissimilar parallel machines. It is shown that this problem can be modeled as an MILP, a CP, a combined MILP-CP OPL model (Van Hentenryck 1999), and a hybrid MILP/CP model. The computational performance of these models for several sets shows that the hybrid MILP/CP model can achieve two to three orders of magnitude reduction in CPU time.
(*Integer Programming; Benders-Decomposition; Constraint Programming; MILP/CP Hybrid Algorithms; Parallel Machine Scheduling*)

1. Introduction

Recently, there has been a significant interest in developing models and methods that combine Mixed Integer Linear Programming (MILP) (Nemhauser and Wolsey 1988) and Constraint Programming (CP) (Marriot and Stuckey 1998) to solve combinatorial optimization problems. The primary reason for this interest is that even though these methodologies can solve similar problems, they have proved to be successful in solving complementary classes of problems. MILP methods have been successfully applied to solve diverse problems, such as network synthesis, crew scheduling, planning, and capital budgeting, that can be modeled as optimization problems. CP methods have proved to be successful in solving highly constrained discrete optimization and feasibility problems for scheduling, configuration, and resource allocation. The main objective of developing integrated models and methods is to use the complementary strengths of MILP and CP for solving problems that are otherwise intractable using either of these two methods alone. In this paper we propose algorithms that use complementary MILP and CP models to achieve this goal for a certain class of optimization problems.

This paper is structured as follows. In the next section, we present a brief background on MILP and CP for solving optimization problems. It is followed by a literature review on integration of these techniques. We then describe a class of problems in which only a subset of the binary variables appears in the objective function of the MILP formulation.

0899-1499/01/1304/0258$5.00
1526-5528 electronic ISSN

We formulate this class of problems as hybrid MILP/CP models that involve some of the MILP constraints, a reduced set of the CP constraints, and the equivalence relations between the MILP and the CP variables. We then propose decomposition and branch-and-bound algorithms to solve these hybrid models. Both of these algorithms rely on relaxed MILP and feasibility CP problems. The aim of these methods is to combine the strength of MILP for proving optimality by using the LP relaxations, and the power of CP for finding feasible solutions by using the specialized propagation algorithms. As an example, we consider a scheduling problem that involves finding a least-cost schedule to process a set of orders using dissimilar parallel machines subject to release and due-date constraints. It is shown that this problem can be modeled as an MILP, CP, or a combined MILP-CP OPL model (Van Hentenryck 1999). We then investigate the computational performance of these alternative models for a number of data sets and highlight the advantages and disadvantages of these approaches. This problem is then modeled as a hybrid MILP/CP model and is solved using the proposed decomposition algorithm. Computational results are presented and finally some conclusions are drawn.

2. Background

MILP-based methods were developed over the last four decades by the Operations Research community (Nemhauser and Wolsey 1988). CP-based methods, on the other hand, are the result of research by the Artificial Intelligence community in the area of Logic Programming and Constraint Satisfaction (Colmerauer 1985, Van Hentenryck 1989, Tsang 1993). Both these frameworks rely on branching to explore the search space. The primary difference lies in the way inference is performed at each node. Linear Programming (LP)-based branch-and-bound methods for MILP involve solving LP subproblems that are generated by dropping some of the constraints (integrality constraints) to obtain bounds on the objective-function values and to prove that a set of constraints is inconsistent. A node is fathomed either when the objective-function value of the LP relaxation is worse than the best integer solution obtained so far, or the LP subproblem is infeasible. Branching is performed whenever the solution obtained by solving the LP relaxation does not satisfy all the constraints in the original problem. If the relaxed solution satisfies all the constraints in the original problem and is better than the best feasible solution found so far, then the best feasible solution is updated. The search terminates when it is proved that no better solution exists than the best feasible solution found.

The effectiveness of MILP methods depends on the size of the linear-programming subproblems, and more importantly, on the gap between the objective for the best feasible solution (optimum) and the objective function value obtained from the initial LP subproblem. There are a number of more sophisticated algorithms that focus on these aspects and use different ways of generating LP subproblems, like Branch and Cut (Padberg and Rinaldi 1991, Balas et al. 1996), and Branch and Price (Barnhart et al. 1998). These algorithms make use of valid inequalities to improve the performance of the solution algorithms.

CP, on the other hand, uses constraint propagation as the inference engine. At each node, constraint propagation is used to reduce the domains of all the variables. The domain of a variable can be continuous, discrete, boolean etc. Constraint propagation can result in empty domains in which case a node is fathomed. Branching is performed whenever the domain of a variable consists of more than one element (discrete and boolean domains) or the bounds are not within a certain tolerance (continuous domains). CP was originally developed to solve feasibility problems. It has now been extended to solve optimization problems. This is achieved by solving a feasibility problem in which the objective function of the problem is rewritten as a constraint that forces it to be equal to a new variable. The domain of this new variable gives upper and lower bounds on the objective-function values. Whenever a feasible solution is obtained during the search, additional constraints that restrict the objective-function values are imposed throughout the search tree. The search terminates when all the nodes have been fathomed. The effectiveness of the CP methods depends primarily on the constraint-propagation algorithms that are used to reduce the domain of a variable.

MILP has the advantage that the effect of all the constraints is evaluated simultaneously and it has a "Global Perspective" on all the constraints (Rodosek et al. 1999). CP, on the other hand, evaluates the effect of constraints sequentially by communicating through domains of the variables. When problems are loosely constrained, finding the optimal solution with CP may prove to be very difficult. However, since MILP requires solving an LP subproblem at each node of the search tree, all the constraints must be linear equalities or inequalities. This imposes a severe restriction on the expressiveness of MILP as a modeling language because for some problems, for example the progressive party problem (Smith et al. 1997), modeling may require a very large number of variables and constraints. Since CP uses constraint propagation instead, it imposes no such restriction. Commercial CP software packages like the ILOG Solver (ILOG 1999d), CHIP (Dincbas et al. 1988), ECLiPSe (Wallace et al. 1997), and Prolog IV allow a number of different constructs over and above algebraic constructs $(+, -, *, =, \leq, \geq)$ that can be used to write constraints in a compact manner. However, caution and sound judgment should be used when exploiting the advantage of having more constructs with which to model the problem. This is because the success of the CP solution approach is highly dependent on the propagation mechanism behind constraints written using these constructs (Heipcke 1999b). Even though more constructs are available, not all of them have efficient constraint-propagation engines. For some applications like sequencing and scheduling, such constraints have proved to be very efficient because strong propagation algorithms are available. It is interesting to point out that for constraints that use only algebraic constructs and use variables with continuous and finite domains, linear programming is sometimes used as the constraint-propagation engine in CP.

3. Literature Review

Recently, a number of papers have compared the performance of CP- and MILP-based approaches for solving a number of different problems, for example the modified generalized assignment problem (Darby-Dowman et al. 1997), the template design problem (Proll and Smith 1998), the progressive party problem (Smith et al. 1997), and the change problem (Heipcke 1999a). Properties of a number of different problems were considered by Darby-Dowman and Little (1998), and their effect on the performance of CP and MILP approaches were presented. As discussed earlier, these papers showed that MILP is very efficient when the relaxation is tight and the models have a structure that can be effectively exploited. CP works better for highly constrained discrete optimization problems where expressiveness of MILP is a major limitation.

Most of the recent attempts (Rodosek et al. 1999, Heipcke 1999b) to integrate CP and MILP use constraint propagation along with linear programming in a single search tree to obtain bounds on the objective and to reduce the domains of the variables. In these approaches a complete CP model and at the least a corresponding partial MILP model are required. This is because CP is a richer modeling tool and not all CP constraints may be easily reformulated as MILP constraints. These approaches in some sense perform redundant computations because a constraint-propagation problem and a simplex problem are solved at every node. For some problems this may be justified because they are intractable for either of the two methods. Rodosek et al. (1999) presented a systematic approach for transforming a CP model into a corresponding MILP model. However, automatic translation from a CP model to an MILP model may result in a poor model involving numerous big-M constraints (poor LP relaxations). In this case, the advantage of performing "Global Reasoning" using LP relaxation is essentially lost. If automatic translation is not used, then the user has to model the problems for both approaches.

Recently, Hooker et al. (1999) argued that a new modeling paradigm may be required to perform efficient integration of MILP- and CP-based approaches. The modeling framework is motivated by the Mixed Logic/Linear modeling framework that was proposed by Hooker and Osorio (1999). Ottosson et al. (2001) presented algorithms for solving such models. For a production-planning problem, they showed that the computational performance of the proposed method

vis-a-vis pure MILP and CP approaches was significantly better. Bockmayr and Kasper (1998) did an interesting analysis of CP and MILP approaches, and presented a unifying framework, *Branch and Infer*, that can be used to develop various integration strategies. They divide constraints for both MILP and CP into two different categories, *primitive* and *non-primitive*. Primitive constraints are those for which there exists a polynomial-time solution algorithm, and non-primitive constraints are those for which this is not true. The interesting aspect about this classification is that some of the primitive constraints in CP are non-primitive in MILP and vice versa. They also discussed how non-primitive constraints can be used to infer primitive constraints and the use of symbolic constraints for MILPs. Raman and Grossmann (1993, 1994) earlier modeled discrete/continuous optimization problems with disjunctions and symbolic constraints in the form of logic propositions. This model, which they denoted as a Generalized Disjunctive Program (GDP), can be converted all or in part into an MILP. They presented the idea of w-MIP representability, which is similar to the idea of primitive constraints. They showed that it is computationally efficient to transform w-MIP representable disjunctions into linear constraints, and proposed a hybrid branch-and-bound algorithm that handles the non w-MIP representable disjunctions directly.

From the work that has been performed, it is not clear whether a general integration strategy will always perform better than either a CP or an MILP approach by itself. This is especially true for the cases where one of these methods is a very good tool to solve the problem at hand. However, it is usually possible to enhance the performance of one approach by borrowing some ideas from the other. For example, Raman and Grossmann (1993) used logic cuts that were written as logic propositions to improve the performance of MILP models. Ideas on edge-finding (Carlier and Pinson 1989, Applegate and Cook 1991) that were used for guiding the search in MILPs to solve jobshop problems, were exploited by Caseau and Laburthe (1994, 1996) and Le Pape (1994) to develop efficient inference engines for scheduling algorithms in CP. Furthermore, there are a number of similarities in some of the underlying ideas of both approaches. For example, probing and integer preprocessing in MILP is in some ways similar to constraint propagation. Chandru and Hooker (1999) give an interesting operations-research perspective on consistency methods and logical inference. Also, Hooker (2000) deals with the subject of MILP and CP integration in detail.

4. Theory

The algorithms proposed in this section have been motivated by the work of Bockmayr and Kasper (1998). These algorithms are based on the premise that combinatorial problems may sometimes have some characteristics that are better suited for MILP and others that are better handled by CP. For these problems, pure MILP- and pure CP-based approaches may not perform well. As discussed earlier, most of the prior work on integrating the two approaches use at least one of the models in complete form (usually CP). In the algorithms that we present the problem is solved using relaxed MILP and CP feasibility models.

Consider a problem, which when modeled as an MILP, has the following structure,

$$\text{(M1): } \min \ c^T x \tag{1}$$

$$s.t. \ \mathbf{A}x + \mathbf{B}y + \mathbf{C}v \leq a \tag{2}$$

$$\mathbf{A}'x + \mathbf{B}'y + \mathbf{C}'v \leq a' \tag{3}$$

$$x \in \{0,1\}^n, y \in \{0,1\}^m, v \in \mathscr{R}^p \tag{4}$$

This is an optimization problem that has both continuous (v) and binary (x and y) variables, and only some of the binary variables (x) have non-zero objective-function coefficients. The constraint set can be divided into two subsets. The first set of constraints (2) models some aspects of the problem that can be represented efficiently in the MILP framework (e.g. assignment constraints) and has a significant impact on the LP relaxation. The second set of constraints (3), on the other hand, is assumed not to affect the LP relaxation significantly, and is sometimes large in number because of the limited expressive power of MILP methods.

The same problem can also be modeled as a CP. Note that more constructs are available in the

CP framework to model the problem (e.g. logical constraints, disjunctions, all-different operator etc; Marriot and Stuckey 1998). For this reason, the MILP and CP models of the same problem may have different variable definitions and constraint structures. However, an equivalence can be established between the constraints and a complete labeling of variables can be derived in one framework from the values of variables in the other one. Let us assume that the equivalent CP model is

$$\text{(M2)}: \min \quad f(\bar{x}) \tag{5}$$

$$s.t. \quad G(\bar{x}, \bar{y}, \bar{v}) \leq 0 \tag{6}$$

$$\bar{x}, \bar{y}, \bar{v} \in \mathscr{D} \tag{7}$$

where $\bar{x}$, $\bar{y}$, and $\bar{v}$ are the CP variables. The domain of these variables ($\mathscr{D}$) can be continuous, discrete, or boolean. Generally, these variables do not have a one-to-one correspondence with the MILP variables (x, y, v), although a mapping between the sets of variables x and $\bar{x}$ can be established. This is because these variables are needed to calculate the objective function. It may or may not be the case for the variables (y,v) and $(\bar{y},\bar{v})$. Usually, equivalence can also be established between the sets of constraints. Let us consider a class of problems with the above mentioned MILP and CP model structures. Let us assume that it is difficult to solve this problem as an MILP because there are a large number of constraints in the constraint set (3) and finding feasible solutions for them is hard. Let us assume that the broader expressive power of CP results in the smaller constraint set (6). Even though the constraint set in CP is much smaller, it may still not be efficient to solve the problem using CP because finding an optimal solution and proving optimality can be difficult for CP (lack of linear-programming relaxations). Ideally, we would like to combine the strength of MILP to handle the optimization aspect of the problem by using LP relaxation and the power of CP to find feasible solutions by using better constraint formulations. To achieve this goal we present a hybrid model for this class of problems that can be solved using either a decomposition algorithm or a branch-and-bound algorithm. The main advantage of the proposed methods is that smaller LP and CP subproblems are solved.

The hybrid model involves MILP constraints, CP constraints, and equivalence relations. The objective function in the hybrid model (M3) is the same as in the MILP model (M1). The constraints for this problem include MILP constraints (2), equivalence relations that relate MILP variables x to CP variables $\bar{x}$, and a reduced set of CP constraints that are derived from the CP constraint set (6) by assuming that the set of CP variables $\bar{x}$ is fixed.

$$\text{(M3)}: \min \quad c^T x \tag{8}$$

$$s.t. \quad \mathbf{A}x + \mathbf{B}y + \mathbf{C}v \leq a \tag{9}$$

$$x \Leftrightarrow \bar{x} \tag{10}$$

$$\overline{G}(\bar{x}, \bar{y}, \bar{v}) \leq 0 \tag{11}$$

$$x \in \{0, 1\}^n, y \in \{0, 1\}^m, v \in \mathscr{R}^p \tag{12}$$

$$\bar{x}, \bar{y}, \bar{v} \in \mathscr{D} \tag{13}$$

It should be noted that the hybrid model (M3) requires at least some variables (x) of the MILP model (M1) and all the variables $(\bar{x}, \bar{y}, \bar{v})$ of the CP model (M2). The values of the CP variables obtained using the model (M3) will always satisfy all the constraints of the CP model (M2). Furthermore, the optimal solution for the problem at hand is given by the values of the CP variables $(\bar{x}, \bar{y}, \bar{v})$ obtained by solving the hybrid model (M3) to optimality. It should be noted that the values of the MILP variables (y,v) obtained from the model (M3) may not be valid for the Model (M1) because the model (M3) does not include the MILP constraint set (3).

An integrated approach is proposed in this work to solve the hybrid MILP/CP model (M3). The basis for this integrated approach are a relaxed MILP problem and a CP feasibility problem (which can in principle be also solved with alternative methods), both of which can be solved efficiently. The relaxed MILP model is used to obtain a solution that satisfies the constraint sets (9) and (12) and optimizes the objective function (8). The solution obtained for the relaxed MILP is used to derive a partial CP solution by using the equivalence relation (10). A CP feasibility model then verifies whether this solution can be extended to a full-space solution that satisfies the constraints (11)

and (13) of the model. If the partial solution from the MILP can be extended, then the full-space solution obtained will also have the same value of the objective function. In this paper we present two methods to search the solution space and obtain the optimal solution. Both of these ideas are essentially the same and use the same relaxed MILP and CP feasibility models. However, the difference lies in the order in which the CP subproblems are solved.

4.1. MILP/CP-Based Decomposition Method

This algorithm has some similarities to Generalized Benders Decomposition (Geoffrion 1972). The algorithm is summarized in Figure 1. In this method a *relaxed MILP model* of the problem, with (8) as the objective and (9 and 12) as constraints, is solved to optimality. Note that integrality constraints on the variable y can be dropped if that makes the relaxed MILP problem easier to solve. If there is no solution, then the original problem is infeasible. Otherwise, values of x are used to determine values of the equivalent CP variables $\bar{x}$. A *CP feasibility problem* then tries to extend this partial solution to a complete solution $(\bar{x}, \bar{y}, \bar{v})$ that satisfies the constraints (11) and (13). If there exists a feasible solution, then this solution is the optimal solution of the problem and the search is terminated. Otherwise the causes for infeasibility are inferred as cuts and added to the relaxed MILP model of the problem. These cuts could be general "no good" cuts (Hooker et al. 1999). In the context of the model (M3) these cuts have the following form for iteration k,

$$\sum_{i\in T^k} x_i - \sum_{i\in F^k} x_i \le B^k - 1 T^k = \{i | x_i^k = 1\}, \qquad (14)$$
$$F^k = \{i | x_i^k = 0\}, B^k = |T^k|$$

Here, $x^k = [x_1^k, x_2^k, \ldots]$ represents the optimal values of x in iteration k. These general "no good" cuts may be rather weak. For this reason, whenever possible stronger cuts ($Q^k x \le q^k$) that exploit the special structure of the problem should be used. The problem-specific cuts should not only cut off the current partial solution, but also eliminate partial solutions with similar characteristics. Cuts are a means of communication between the relaxed MILP and CP feasibility models and play a very important role in the success of these methods. The entire procedure is repeated until the solution obtained using the relaxed MILP model can be extended to a full feasible solution, or the relaxed MILP problem becomes infeasible.

Assuming that the general "no good" cuts (14) are used we can establish the following convergence proof for the MILP/CP-based decomposition method.

Theorem 1. *If the MILP/CP decomposition method is applied to solve problem* (M3) *with the cuts in* (14), *then the method converges to the optimal solution or proves infeasibility in a finite number of iterations.*

Proof. The problem (M3) is not unbounded because the domain of the variable x is bounded. So the problem can either be fesible or infeasible. Let us consider each of the cases individually.

Case 1: Problem (M3) is feasible

The relaxed MILP master problem with cuts in (14) after iteration K is given by

$$\begin{aligned}
(\mathrm{RM}^K)\ \min\ & z = c^T x \\
s.t.\ & \mathbf{A}x + \mathbf{B}y + \mathbf{C}v \le a \\
& \sum_{i\in T^k} x_i - \sum_{i\in F^k} x_i \le B^k - 1 \quad \forall k \in \{1, 2, \ldots, K-1\} \\
& T^k = \{i | x_i^k = 1\}, F^k = \{i | x_i^k = 0\}, B^k = |T^k| \\
& \qquad \forall k \in \{1, 2, \ldots, K-1\} \\
& x \in \{0,1\}^n, y \in \{0,1\}^m, v \in \mathscr{R}^p
\end{aligned}$$

Let x^K be the optimal solution for this relaxed MILP master problem. The corresponding CP feasibility subproblem is given by,

$$\begin{aligned}
(\mathrm{CP}^K)\ \text{Find}\ & \bar{y}, \bar{v} \\
s.t.\ & \bar{G}(\bar{x}^K, \bar{y}, \bar{v}) \le 0 \\
& \bar{y}, \bar{v} \in \mathscr{D}
\end{aligned}$$

where ($x^K \Leftrightarrow \bar{x}^K$). Note that for all iterations $k < K$, the relaxed MILP master problems (RM^k) are feasible and the (CP^k) subproblems are infeasible. If either of the two conditions are not satisfied the algorithm terminates before reaching iteration K. Since the master problem for the first iteration is a relaxation of the

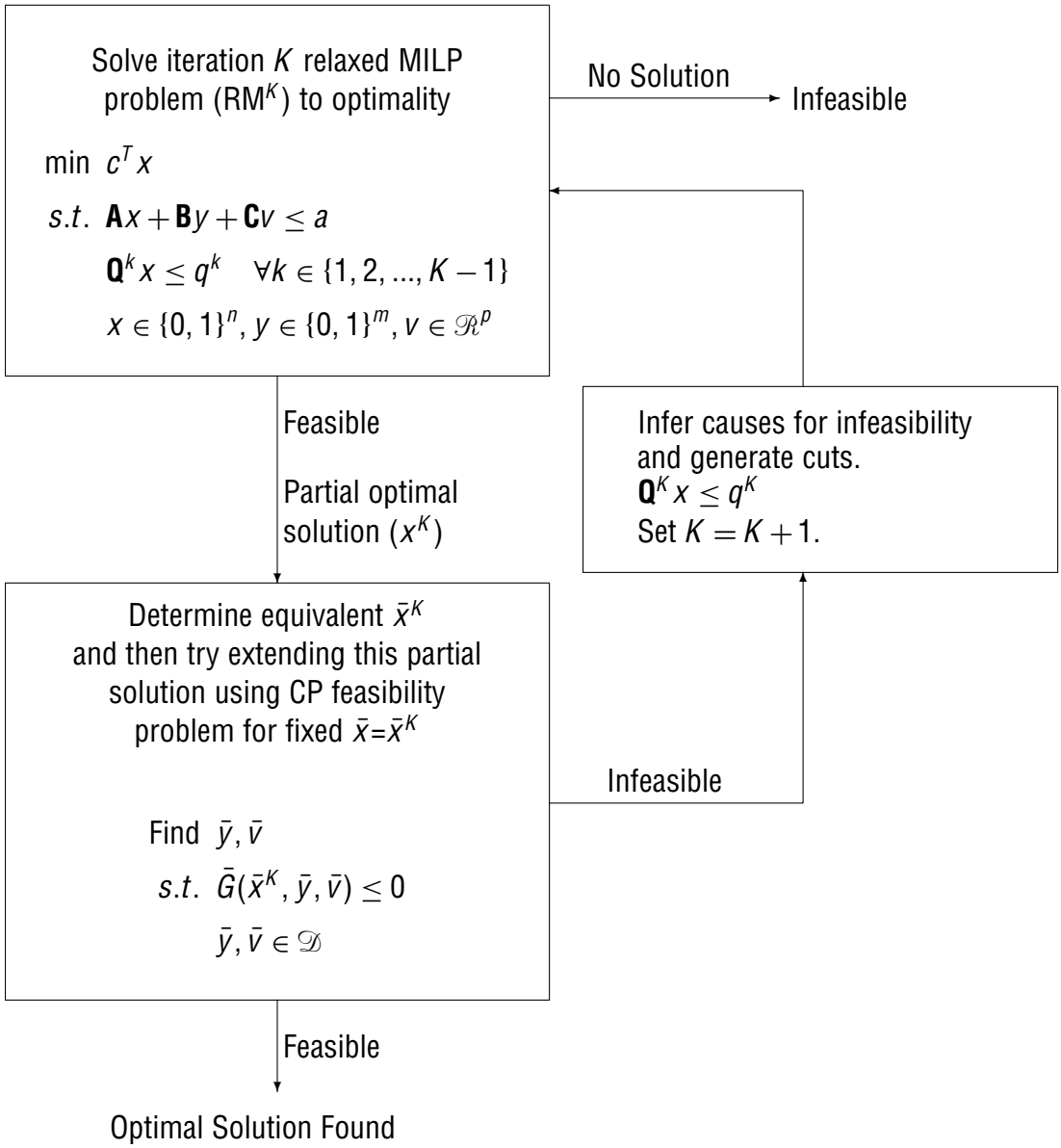

Figure 1 **MILP/CP Decomposition Method**

original problem, the MILP problem (RM^K) is also a relaxation of the problem (M3). This because the "no good" cuts that have been added until iteration K do not exclude any feasible solution from the original problem (M3). Furthermore, the domain of successive relaxation is strictly smaller than the previous one. Since the domain of x is finite (in the worst case has 2^n elements) and is non empty for problem (M3), it implies that the algorithm will reach a step where the (CP^k) subproblem is feasible. Assume that at iteration K, the subproblem (CP^K) yields a feasible solution $(\bar{x}^K, \bar{y}, \bar{v})$. This solution is optimal for problem (M3) as it belongs to the domain of (M3) and the objective-function value for this solution equals the optimum

objective-function value ($c^T x^K$) obtained for a relaxation of problem (M3). Hence, if there is a feasible solution to problem (M3), the algorithm will converge to an optimal solution in a finite number of iterations.

Case 2: Problem (M3) is infeasible

If the problem is infeasible the subproblem (CP^k) will never result in a feasible solution. Furthermore, the domain of x is finite and with each successive iteration it shrinks by one integer point. Hence, after a finite number of steps the master problem becomes infeasible, proving infeasibility in a finite number of steps. □

There are two ways in which this method can be implemented. The easier, but less efficient approach that was used in this paper, is to solve the relaxed MILP model from scratch whenever the cuts are updated. A more efficient approach will be to store the branch-and-bound tree for the relaxed MILP problem and update it whenever cuts are added (e.g. see Quesada and Grossmann 1992).

4.2. LP/CP-Based Branch-and-Bound Method

This algorithm extends the basic branch-and-bound (B&B) algorithm for mixed integer problems to solve problems that are represented using the hybrid MILP/CP model (M3). The idea is in principle straightforward, although it may be difficult to implement. In the B&B algorithm, the current best integer solution is updated whenever an integer solution with an even better objective-function value is found. In the proposed algorithm an additional CP feasibility problem is solved to ensure that the integer solution obtained for the relaxed MILP problem can be extended in the full space. It is only in this case that the current best integer solution for the relaxed MILP problem is updated. If it cannot be extended, then the best current integer solution is not updated and cuts are added to the current and all other open nodes.

The proposed B&B method involves solving a series of LP subproblems obtained by branching on the integer variables. An LP subproblem p for the proposed algorithm has the form

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & \mathbf{A}x + \mathbf{B}y + \mathbf{C}v \le a \\ & \mathbf{Q}x \le q \quad \text{Cuts} \\ & x_p^{LB} \le x \le x_p^{UB} \\ & y_p^{LB} \le y \le y_p^{UB} \\ & x \in \mathscr{R}^n,\ y \in \mathscr{R}^m,\ v \in \mathscr{R}^p \\ & x_p^{LB}, x_p^{UB}, y_p^{LB}, y_p^{UB} \in \{0, 1\} \end{aligned}$$

The difference in various LP subproblems is only in the upper and lower bounds for all the integer variables. Also, the set of cuts is updated as the search progresses. The objective-function value for any feasible solution of the problem in the full space provides an upper bound (UB) of the objective function. Let P denote the set of LP subproblems to be solved. The LP/CP B&B Method can be summarized as follows:

1. Initialization. $UB = \infty$, $P = \{p^0\}$. LP subproblem p^0 is generated by using the same lower and upper bounds on the integer variables as the original problem, and it does not include any cuts.

2. Check if there are any more problems to be solved. If $P = \varnothing$ then go to step 5; else go to 3.

3. Fathom a Node. Select and remove an LP subproblem p from the set P. The criterion for selecting an LP is also called a *node selection rule*. There are many different rules for selecting an LP, and they play a key role in the efficiency of the B&B algorithm (Nemhauser and Wolsey 1988).

Solve LP subproblem p.

- If the LP is infeasible or the optimal value of the objective function (lower bound) is greater than UB then go to step 2.
- If any one of the integer variables does not have integral values then go to step 4.
- If the solution has integral values for all the integer variables then determine the CP variables $\bar{x}$ using the values of LP variables x. For fixed $\bar{x}$ solve the following CP feasibility problem that tries to extend this partial solution.

$$\begin{aligned} \text{Find} \quad & \bar{v}, \bar{y} \\ s.t. \quad & \overline{G}(\bar{x}, \bar{y}, \bar{v}) \le 0 \\ & \bar{y}, \bar{v} \in \mathscr{D} \end{aligned}$$

If the CP problem is feasible then update UB; otherwise, add the cuts in (14) or infer the causes for infeasibility to generate tighter cuts and add them to all

the LP subproblems belonging to set P by updating the set (Q, q). Go to step 2.

4. Branch on a Variable. Of all the variables that have been assigned non-integral values, select one according to some prespecified *branching variable selection rule* (Nemhauser and Wolsey 1988). Let us denote this integer variable by z_p. Generate two LP subproblems p^1 and p^2 and add them to set P. The subproblem p^1 is generated by specifying the floor of the optimal value of z_p as the upper bound for z_p, and the subproblem p^2 is generated by specifying the ceiling of the optimal value of z_p as the lower bound of z_p. Go to step 2.

5. Termination. If $UB = \infty$, then the problem does not have a feasible solution or it is unbounded. Otherwise, the optimal solution corresponds to the current value UB.

For the proposed decomposition and B&B methods to be successful, it is of course very important to choose suitable relaxed MILP and CP feasibility models. Furthermore, rather than using the cuts in (14) inferring strong cuts, $Qx \leq q$, as causes for infeasibility of CP feasibility models can significantly reduce the number of problems to be solved. These cuts, however, are non-trivial and should be derived whenever possible for each class of problems at hand. It is also worth emphasizing that the proposed algorithms do not impose any restriction on the structure of the equivalence relation in (10). It can even be procedural. It should be noted that hybrid models similar to the ones presented in this section can also be solved using OPL (Van Hentenryck 1999). In OPL, the solution algorithm for the hybrid model solves an LP subproblem involving all the linear constraints, as well as a constraint-propagation subproblem involving all the constraints at every node of the search tree. Usually, all the original CP constraints (6) are needed in the hybrid MILP-CP OPL model. Furthermore, the equivalence relations (10) must be written in closed form as equations, inequalities, or symbolic relations.

Even though the algorithms presented in this section are limited to the MILP models that have only a subset of binary variables with non-zero coefficients in the objective function, they can in principle be generalized for MILP models that have both binary and continuous variables in the objective function. The partial solution will then involve both binary and continuous variables and the CP problem will extend this partial solution in the full space. However, in such a case the biggest challenge is to derive effective cuts that exclude partial solutions that are feasible for the master problem but can not be extended in the full space.

5. Scheduling Problem

We consider a specific scheduling problem that falls into the class of problems considered in this paper and that is similar to the one considered by Hooker et al. (1999). This scheduling problem involves finding a least-cost schedule to process a set of orders I using a set of dissimilar parallel machines M. Processing of an order $i \in I$ can only begin after the release date r_i and must be completed at the latest by the due date d_i. Order i can be processed on any of the machines. The processing cost and the processing time of order $i \in I$ on machine $m \in M$ are C_{im} and p_{im}, respectively. In this section, we consider three alternative strategies to model and solve this particular problem.

5.1. MILP Model

The scheduling problem described above can be modeled as an MILP. The main decisions involved in this scheduling problem are assignment of orders on machines, sequence of orders on each machine, and start time for all the orders. The binary variable x_{im} is an assignment variable, and it is equal to one when order i is assigned to machine m. Binary variable $y_{ii'}$ is the sequencing variable, and it is equal to one when both i and i' are assigned to the same machine and order i' is processed after order i. The continuous variable ts_i denotes the start time of order i. Using these variables the MILP model for this problem can be written as

$$\min \sum_{i \in I} \sum_{m \in M} C_{im} x_{im} \tag{15}$$

$$\text{s.t.} \quad ts_i \geq r_i \quad \forall i \in I \tag{16}$$

$$ts_i \leq d_i - \sum_{m \in M} p_{im} x_{im} \quad \forall i \in I \tag{17}$$

$$\sum_{m \in M} x_{im} = 1 \quad \forall i \in I \tag{18}$$

$$\sum_{i\in I} x_{im}p_{im} \le \max_i\{d_i\} - \min_i\{r_i\} \quad \forall m \in M \tag{19}$$

$$y_{ii'} + y_{i'i} \ge x_{im} + x_{i'm} - 1 \quad \forall i, i' \in I, i' > i, m \in M \tag{20}$$

$$ts_{i'} \ge ts_i + \sum_{m\in M} p_{im}x_{im} - U(1 - y_{ii'}) \quad \forall i, i' \in I, i' \ne i \tag{21}$$

$$y_{ii'} + y_{i'i} \le 1 \quad \forall i, i' \in I, i' > i \tag{22}$$

$$y_{ii'} + y_{i'i} + x_{im} + x_{i'm'} \le 2 \quad \forall i, i' \in I, i' > i, m, m' \in M, m \ne m' \tag{23}$$

$$ts_i \ge 0 \tag{24}$$

$$x_{im} \in \{0, 1\} \quad \forall i \in I, m \in M \tag{25}$$

$$y_{ii'} \in \{0, 1\} \quad \forall i, i' \in I, i' \ne i \tag{26}$$

Objective (15) of this problem is to minimize the processing cost of all the orders. Constraints (16) and (17) ensure that processing of an order i starts after the release date and is completed before the due date. An order needs exactly one machine for processing and this is enforced using assignment constraint (18). Inequality (19) is a valid cut, and it tightens the LP relaxation of the problem. It is based on the fact that the total processing time of all the orders that are assigned on the same machine should be less than the difference of the latest due date and the earliest release date. Constraint (20) is a logical relationship between assignment and sequencing variables. The underlying logic behind this constraint is that if order i and i' are assigned to machine m then they must be processed one after the other. Constraint (21) is the sequencing constraint in which $U = \sum_{i\in I} \max_{m\in M}\{p_{im}\}$. It ensures that if the sequencing variable $y_{ii'}$ is one, then order i' is processed after order i. Constraints (22) and (23) are logical cuts. The former is based on the logic that either i is processed before i' or vice versa. The latter ensures that sequencing variables are zero if i and i' are assigned to different machines. Adding both of these constraints reduces the computational time needed to solve the MILP very significantly. Other logic cuts were also tried, but they were not as effective as (22) and (23).

5.2. CP Model

The same scheduling problem can also be modeled using CP. The CP model, in contrast to an MILP model, is highly dependent on the CP package used to model the problem because of the differences in constructs available in various modeling languages. In this paper we use ILOG's OPL modeling language (Van Hentenryck 1999). OPL has a set of constructs especially designed for scheduling problems that can be used to develop a compact CP model for the scheduling problem at hand. We will not go into the details of the modeling language, but will describe the constructs that have been used to model our scheduling problem. The basic OPL modeling framework involves a set of orders that need to be completed using a certain set of resources. A number of different resource types are available to capture the nature of the problem. For our scheduling problem the parallel machines are the resources, and they can be modeled as *unary resources* in OPL. The defining attribute of a *unary resource* is that it can process only one order at any instance. Each order is associated with a start date and duration. Using this basic framework a CP model using OPL constructs can be written as follows:

$$\min \sum_{i\in I} C_{iz_i} \tag{27}$$

$$\text{s.t.} \quad i.\text{start} \ge r_i \quad \forall i \in I \tag{28}$$

$$i.\text{start} \le d_i - p_{z_i} \quad \forall i \in I \tag{29}$$

$$i.\text{duration} = p_{z_i} \quad \forall i \in I \tag{30}$$

$$i \text{ requires } T \quad \forall i \in I \tag{31}$$

$$\text{activityHasSelectedResource}(i, T, t_m) \Leftrightarrow z_i = m \quad \forall i \in I, m \in M \tag{32}$$

$$z_i \in M \quad \forall i \in I \tag{33}$$

$$i.\text{start} \in \mathcal{I} \quad \forall i \in I \tag{34}$$

$$i.\text{duration} \in \mathcal{I} \quad \forall i \in I \tag{35}$$

Here z_i is a variable subscript and represents the machine selected to process order i. Variable subscripts are powerful modeling constructs and available in most CP software packages. The variable subscript z_i has been used in the objective function (27) to calculate the cost of order i (C_{iz_i}) directly. Constraints (28) and (29) ensure that processing of order i begins after release date r_i and is completed before the due date d_i. The start time of the order is

given by "i.start". The duration of an order depends on the machine used to process it. This is enforced in constraint (30), which uses the variable subscript z_i to calculate processing time of order i. Constraint (31) uses a special OPL construct "requires". It enforces that order i needs a *unary resource* from the set of *unary resources* T. Note that this constraint enforces the assignment of an order to a specific machine as well as the sequencing of orders that have been assigned to the same machine. Constraint (32) uses another OPL-specific boolean function "activityHasSelectedResource()" that returns a value of true or false. In the context of our scheduling problem this function returns a value of "true" if order i is processed using the *unary resource* corresponding to the machine m ($t_m \in T$), and false otherwise. Constraint (32) ensures that if order i is processed using the *unary resource* t_m, then the subscript variable z_i is equal to m. It links the decisions made by the inference engine behind the construct "requires" and other constraints in the model.

It can be clearly seen that variable and constraint definitions in the CP model for the scheduling problem at hand are very different from the MILP model. This is primarily because specialized CP constructs were used to model the problem at hand. However, there exists a close resemblance in the two modeling frameworks. The subscript variable z_i in the CP model is equivalent to the binary variable x_{im} in the MILP model. The start time of order i has the same definition in the two models. The sequencing variable $y_{ii'}$ in the MILP model has no equivalent variable in CP model. However, the value of this variable can be obtained from the solution of the CP model. Equivalence can also be drawn between the constraints in the two frameworks. Constraints (28) and (29) are equivalent to (16) and (17), respectively. The set of constraints (30) to (32) play the same role as the set of constraints (18), (20), and (21). One major difference between the two models is that the MILP model is a continuous-time scheduling model. The CP model, on the other hand, is a discrete-time scheduling model (start time is an integer variable in the OPL framework).

5.3. Combined MILP-CP OPL Model

Instead of developing pure MILP and CP models, it is also possible to write a combined model. OPL allows modeling of a problem using a combined MILP-CP model in which constraints are expressed in both forms. Even though the size of the integrated model is larger, it may still perform better in some cases because fewer nodes may have to be explored. The solution algorithm for the integrated model primarily uses the CP solver. However, at each CP node an extra LP relaxation, consisting of all the linear constraints in the combined model, is solved to obtain bounds for the objective function (Van Hentenryck 1999). The combined MILP-CP OPL model for the scheduling problem at hand can be written as:

$$\min \quad \sum_{i\in I}\sum_{m\in M} C_{im}x_{im} \tag{36}$$

s.t.

$$\left.\begin{array}{l} ts_i \geq r_i \quad \forall i \in I \\ ts_i \leq d_i - \sum_{m\in M} p_{im}x_{im} \quad \forall i \in I \\ \sum_{m\in M} x_{im} = 1 \quad \forall i \in I \\ \sum_{i\in I} x_{im}p_{im} \leq \max_i\{d_i\} - \min_i\{r_i\} \\ \qquad \forall m \in M \end{array}\right\} \text{MILP} \tag{37}$$

$$\left.\begin{array}{l} i.\text{start} \geq r_i \quad \forall i \in I \\ i.\text{start} \leq d_i - p_{z_i} \quad \forall i \in I \\ i.\text{duration} = p_{z_i} \quad \forall i \in I \\ i \text{ requires } T \quad \forall i \in I \\ \text{activityHasSelectedResource}(i, T, t_m) \\ \qquad \Leftrightarrow z_i = m \quad \forall i \in I, m \in M \end{array}\right\} \text{CP} \tag{38}$$

$$\left.\begin{array}{l} x_{iz_i} = 1 \quad \forall i \in I \\ i.\text{start} = ts_i \quad \forall i \in I \\ \overline{C}_i = \sum_{m\in M} C_{im}x_{im} \quad \forall i \in I \\ \overline{C}_i = C_{iz_i} \quad \forall i \in I \end{array}\right\} \begin{array}{c}\text{Linking}\\ \text{Constraints}\end{array} \tag{39}$$

$$ts_i \geq 0 \tag{40}$$

$$x_{im} \in \{0, 1\} \quad \forall i \in I, m \in M \tag{41}$$

$$z_i \in M \quad \forall i \in I \tag{42}$$

$$i.\text{start} \in \mathcal{I} \quad \forall i \in I \tag{43}$$

$$i.\text{duration} \in \mathcal{I} \quad \forall i \in I \tag{44}$$

This model is a combination of the pure MILP and pure CP models that were presented earlier. The objective function (36) of the problem is the same as in the MILP model. Constraint set (37) includes all of the MILP constraints except for constraints (20) to (23). There is no theoretical restriction for excluding constraints (20) to (23) per se. The only reason that these constraints were dropped is that they are large in number and do not significantly tighten the LP relaxation of the problem. Constraint set (38) includes all the constraints from the CP model. Finally, constraint set (39) links MILP variables and CP variables. The first constraint in (39) links the assignment variables, the second constraint links the start times, and the rest are extra constraints that make the propagation stronger.

5.4. Computational Results

The scheduling problem can be solved using any one of the three different optimization models that were presented in the previous section. All the three models can be implemented in ILOG OPL Studio 2.1 (ILOG 1999b). It uses CPLEX 6.5 (ILOG 1999a) to solve an MILP, ILOG Solver 4.4 (ILOG 1999d) and ILOG Scheduler 4.4 (ILOG 1999c) to solve a CP, and all of them to solve a combined MILP-CP OPL model. To study the behavior and characteristics of the three models, we consider several numerical example problems.

In this section, we consider instances of varying complexity from the class of scheduling problems presented in the previous section. The size of the optimization models for these problems depends on the number of orders and the number of parallel machines. We consider problems of five different sizes in terms of the number of orders and the number of parallel machines, as shown in Table 1. The parameters for these problems are processing costs (C_{im}), release dates (r_i), due dates (d_i), and processing times (p_{im}). For each problem size, we consider two sets of data. The objective is to demonstrate that the difficulty of solving such a scheduling problem can vary significantly with data. Therefore, we have a total of ten instances of the scheduling problem. Details of these ten problems, five different sizes with two data sets each, are presented in the Online Supplement to this paper (available in the electronic format at the INFORMS Journal on Computing website at http://joc.pubs.informs.org/).

Table 1 Number of Orders and Machines in Each Problem

Problem	Number of Orders	Number of Machines
1	3	2
2	7	3
3	12	3
4	15	5
5	20	5

The only distinguishing characteristic of the two data sets for each problem is that the processing times in the first data set are longer. For this reason, the problems corresponding to this data set have fewer feasible schedules and the total cost of processing all the tasks is higher. The computational results for solving these problems using the MILP model and the CP model are presented in Tables 2 and 3, respectively. It can be clearly seen that for this class of problems the CP models are smaller in size than the MILP models. Furthermore, the size of the MILP model increases much more rapidly in comparison to the CP model. For all the problem sizes, the first data set is more difficult to solve for both the MILP and the CP methods. Also, it took less time to solve the problems using the CP model in comparison to the MILP model. However, the computational effort increases rapidly in both cases. The problem corresponding to the first

Table 2 Computational Results for the MILP Model

Problem	Set	Constraints	Variables	Nodes	Time[a]	Objective
1	1	32	18	0	0.01	26
1	2	32	18	0	0.03	18
2	1	276	77	38	0.47	60
2	2	276	77	54	0.49	44
3	1	831	192	29446	220.01	101
3	2	831	192	180	1.77	83
4	1	2990	315	8891	180.41	115
4	2	2990	315	3855	61.82	102
5	1	5385	520	80000	20000[b]	171[c]
5	2	5385	520	2114	106.28	140

[a]Using CPLEX 6.5 single processor version on a dual processor SUN Ultra 60 workstation
[b]Search terminated because the tree size exceeded 300 MB
[c]Suboptimal solution; lower bound at termination = 156.011019

Table 3 Computational Results for the CP Model

Problem	Set	Constraints	Variables	Failures	Choice Points	Time[a]	Objective
1	1	15	33	3	3	0.00	26
1	2	15	33	15	19	0.02	18
2	1	42	77	12	12	0.04	60
2	2	42	77	172	185	0.14	44
3	1	72	132	4743	4748	3.84	101
3	2	72	132	583	606	0.38	83
4	1	120	165	469918	469938	553.54	115
4	2	120	165	5435	5467	9.28	102
5	1	160	220	55529196	55529254	68853.49[b]	165[c]
5	2	160	220	1307486	1307532	2673.87	140

[a]Using ILOG Scheduler 4.4 and ILOG Solver 4.4 single processor versions on a dual processor SUN Ultra 60 workstation
[b]Computational time for finding the suboptimal solution
[c]Suboptimal Solution

data set of the biggest example (Problem 5) could not be solved to optimality using any of the two methods in reasonable computational time. It should be noted that computational times for MILP model may be possibly reduced further by using more sophisticated schemes like branch and cut. Similarly, computational times for the CP model may be possibly reduced by developing special branching schemes for this example. These strategies are outside the scope of this paper.

The advantage of the MILP model is that the values of the assignment variables obtained from the LP relaxations direct the branch-and-bound search to obtain the least-cost assignments. However, scheduling the orders on each machine is a more daunting task. This is because the sequencing constraint (21) has a big-M form and results in a poor relaxation. Furthermore, the sequencing variables do not directly contribute to the objective-function value and the LP relaxation may or may not direct these variables towards feasibility. On the other hand, the CP methods use effective constraint-propagation algorithms for scheduling that are based on the ideas of edge finding and task intervals (Caseau and Laburthe 1994, 1996). However, an assignment is chosen using a predefined enumeration strategy by excluding the infeasible schedules.

Clearly, there are complementary strengths of the MILP and the CP methods that may possibly be combined to tackle more difficult problems. One of the possible ways is to use the combined MILP-CP OPL model that was presented in the previous section. The computational results for solving the set of ten example problems using the MILP-CP OPL model are presented in Table 4. Clearly, the MILP-CP OPL model for all the problems is larger than the corresponding CP model. However, it is still smaller than the corresponding MILP model because the sequencing variables and constraints were not included in the combined MILP-CP OPL model. It can be clearly seen that all the problems could be solved faster using the combined model. Furthermore, the first data set of the fifth example problem that was intractable for both the MILP and CP methods alone could now be solved to optimality using the combined method.

6. MILP/CP Hybrid Model

The scheduling problem at hand can be posed in the proposed hybrid MILP/CP modeling framework as follows,

$$\min \sum_{i\in I}\sum_{m\in M} C_{im}x_{im} \tag{45}$$

$$\text{s.t.}\quad ts_i \geq r_i \quad \forall i \in I \tag{46}$$

$$ts_i \leq d_i - \sum_{m\in M} p_{im}x_{im} \quad \forall i \in I \tag{47}$$

$$\sum_{m\in M} x_{im} = 1 \quad \forall i \in I \tag{48}$$

$$\sum_{i\in I} x_{im}p_{im} \leq \max_i\{d_i\} - \min_i\{r_i\} \quad \forall m \in M \tag{49}$$

Table 4 Computational Results for the Combined MILP-CP OPL Model

Problem	Set	Constraints	Variables	Failures	Choice Points	Time[a]	Objective
1	1	38	49	3	3	0.04	26
1	2	38	49	15	19	0.05	18
2	1	94	120	13	13	0.10	60
2	2	94	120	175	188	0.27	44
3	1	159	205	1939	1944	4.21	101
3	2	159	205	693	716	1.12	83
4	1	230	286	43882	43902	91.59	115
4	2	230	286	3057	3090	5.58	102
5	1	305	381	4929656	4929686	13736.06	158
5	2	305	381	80290	80337	170.95	140

[a]Using CPLEX 6.5, ILOG Scheduler 4.4, and ILOG Solver 4.4 single processor versions on a dual processor SUN Ultra 60 workstation

$$\text{if } (x_{im} = 1) \text{ then } (z_i = m) \quad \forall i \in I, m \in M \tag{50}$$

$$i.\text{start} \geq r_i \quad \forall i \in I \tag{51}$$

$$i.\text{start} \leq d_i - p_{z_i} \quad \forall i \in I \tag{52}$$

$$i.\text{duration} = p_{z_i} \quad \forall i \in I \tag{53}$$

$$i \text{ requires } t_{z_i} \quad \forall i \in I \tag{54}$$

$$ts_i \geq 0 \tag{55}$$

$$x_{im} \in \{0, 1\} \quad \forall i \in I, m \in M \tag{56}$$

$$z_i \in M \quad \forall i \in I \tag{57}$$

$$i.\text{start} \in \mathcal{I} \quad \forall i \in I \tag{58}$$

$$i.\text{duration} \in \mathcal{I} \quad \forall i \in I \tag{59}$$

The objective function (45) for the hybrid model is the same as the objective function (15). Constraints (46) through (49) are the MILP constraints (16) through (19). These constraints are equivalent to constraint set (9) of model (M3). Constraint (50) establishes the equivalence between the MILP variable x_{im} and the CP variable z_i, and it is represented by (10) in model (M3). It should be noted that there is no need to write the equivalence relation in closed form as equations or inequalities. This is in sharp contrast to the set of equivalence relations (39) used in the MILP-CP OPL model. Finally, constraints (51) through (54) are the reduced set of CP constraints derived from the original CP constraints (28) through (32) by assuming that the assignment of orders to machines has already been made. Note that the CP constraint (32) is no longer needed, and in constraint (54) order i requires the *unary resource* corresponding to the machine to which it has been assigned. These constraints are equivalent to constraint set (11) of model (M3).

Any one of the two proposed algorithms can be used to solve this hybrid MILP/CP model for the scheduling problem at hand. In this section, we use the decomposition algorithm that was presented in Figure 1 for solving the hybrid MILP/CP model. The reason for choosing this method is that it is easy to implement and is sufficient to test the potential usefulness of the proposed methodology. Details of the decomposition algorithm in the context of the proposed hybrid MILP/CP scheduling model are presented in Figure 2. In this algorithm, the relaxed MILP problem assigns a machine to every order. The CP feasibility problem then attempts to find a feasible schedule for this assignment. If a feasible schedule can be obtained, then it is the optimal solution. Otherwise, causes of infeasibilities are inferred as integer cuts to exclude assignments with similar characteristics. The relaxed MILP problem is resolved to obtain another assignment. If no other assignment is possible, then the scheduling problem is infeasible.

Integer cuts used in the decomposition algorithm are very critical to its efficiency. Instead of using the integer cuts in (14), the cuts used for the scheduling problem at hand are based on the idea that if a set of orders cannot be scheduled on a particular machine, then it will not be possible to find a feasible schedule for any assignment in which all those orders are assigned to that machine. Note that a cut is generated

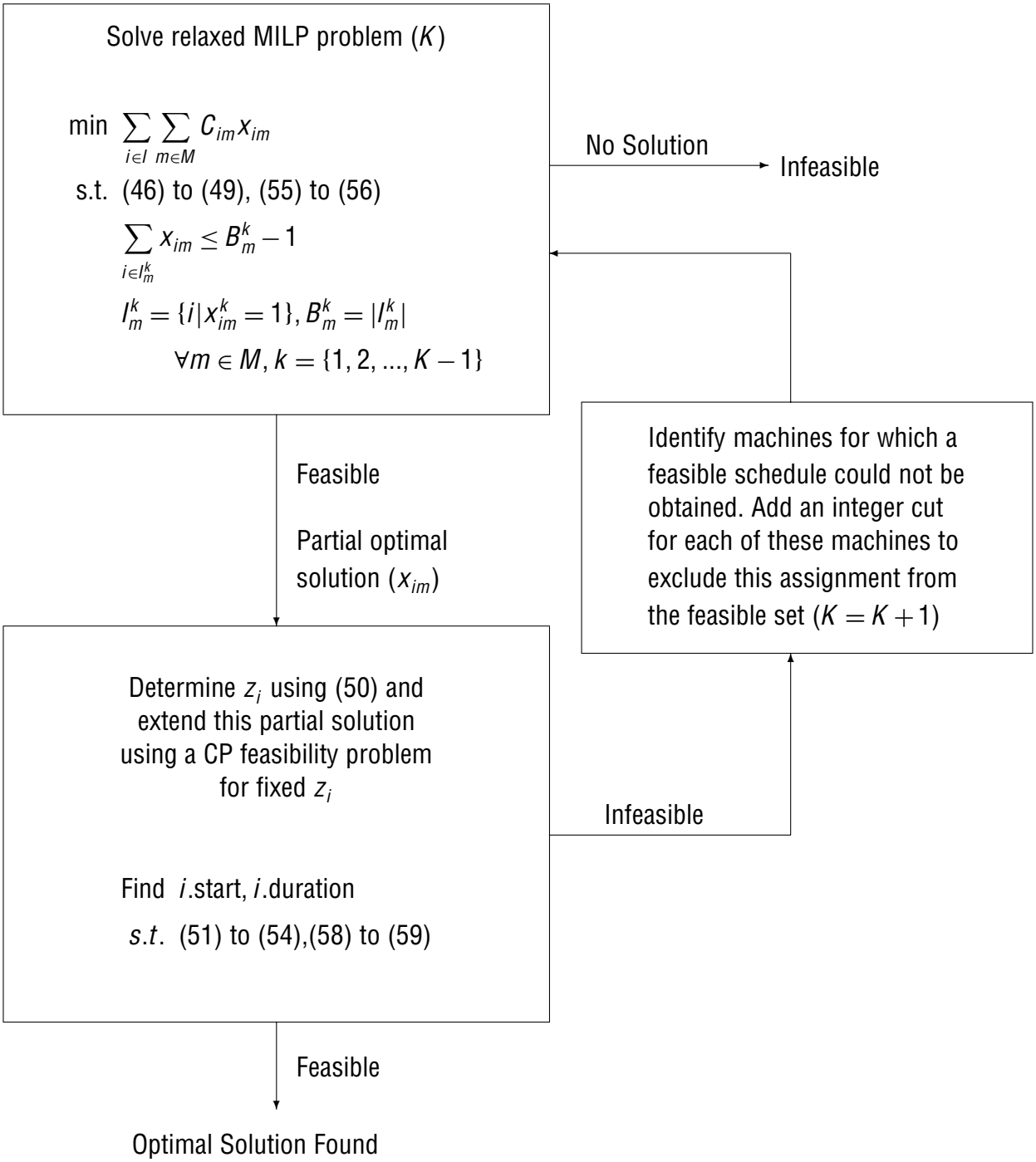

Figure 2 **MILP/CP Decomposition Method for the Scheduling Problem**

for each machine that could not be scheduled successfully. The integer cuts for iteration k, which are stronger than equation (14), have the following general form,

$$\sum_{i \in I_m^k} x_{im} \leq B_m^k - 1 \quad \forall m \in M \tag{60}$$

$$I_m^k = \{i | x_{im}^k = 1\}, B_m^k = |I_m^k|$$

Here x_{im}^k is the optimal value of x_{im} in iteration k. As an example consider a case where orders 1, 2, and 3 were assigned to machine A but could not be scheduled. The corresponding integer cut is

$$x_{1A} + x_{2A} + x_{3A} \leq 2$$

This cut in essence states that any assignment in which orders 1, 2, and 3 are all assigned to Machine A is not allowed. It is worth emphasizing once again that these constraints cut off more that one possible overall assignment. Furthermore, it should be noted that in this scheduling problem parallel machines do not interact. Therefore, once the assignment has been specified, a schedule for order processing can be derived for each machine individually. This means that a CP feasibility problem for $|M|$ parallel machines can be decomposed into $|M|$ CP feasibility problems for a single machine each. The advantage of decomposing the problem in such a manner is that it becomes very easy to identify machines for which a feasible schedule could not be obtained.

The proposed decomposition algorithm was implemented using the scripting language in OPL Studio 2.1 (ILOG 1999b). In our implementation, the relaxed MILP is solved from scratch whenever cuts are added to it. One major iteration of the decomposition algorithm involves solving a relaxed MILP problem and $|M|$ CP feasibility problems. The size of the relaxed MILP problem increases as the search progresses. The size of the CP feasibility problem for each machine depends on the number of orders assigned to that machine, and it varies from one major iteration to another. All ten instances of the scheduling problem at hand that were presented in the earlier section can be solved using the decomposition algorithm. The computational results for them are summarized in Table 5.

In this table, the size of the CP subproblems has not been reported as it varies from one iteration to another, and only the initial size of the relaxed MILP problems has been reported. However, the final number of constraints in the relaxed problem can be obtained by adding the number of cuts to the initial number of constraints. The computational time for solving all the relaxed MILP problems is reported in the seventh column. Note that the computational time reported for solving the CP-feasibility problems is only for the ones that were feasible. This is because in the current implementation of the OPL scripting language it is not possible to obtain the computational time if the CP-feasibility problem is infeasible. In our experience the computational time for solving an infeasible CP subproblem was of the same order of magnitude as solving a feasible CP subproblem. The number of CP-feasibility subproblems solved for a problem is equal to the product of the number of machines and the number of major iterations required. Recall that a cut is added for every infeasible CP subproblem. Hence, the number of CP subproblems for which the computational time was not included is equal to the total number of cuts.

It can be clearly seen that all the instances of the scheduling problem at hand can be solved very efficiently using the proposed decomposition algorithm. For the larger instances of this scheduling problem the computational times were significantly lower compared to any of the three methods that were presented in the previous section. Recall that the first data set of all the five problems was more difficult to solve. This is because fewer assignments lead to a feasible schedule. Therefore, more iterations are needed and more integer cuts had to be added. Based on these computational results we can conclude that the proposed MILP and CP integration strategy can tackle this scheduling problem much better than either MILP or CP, and even the MILP-CP OPL solution approach. Further numerical results for this problem with different data are reported in Harjunkoski et al. (2000).

7. Conclusions

The objective of this paper has been to develop models and methods that use complementary strengths of

Table 5 Computational Results for Hybrid MILP-CP Model

Problem	Set	MILP Size Constraints	MILP Size Variables	Major Iterations	Cuts	MILP Time[a]	CP Time[b]	Objective
1	1	11	9	2	1	0.02	0.00	26
1	2	11	9	1	0	0.01	0.00	18
2	1	24	28	13	16	0.47	0.05	60
2	2	24	28	1	0	0.01	0.01	44
3	1	39	48	31	43	4.01	0.17	101
3	2	39	48	1	0	0.02	0.00	83
4	1	50	90	18	26	2.01	0.24	115
4	2	50	90	1	0	0.02	0.02	102
5	1	65	120	31	60	13.69	0.44	158
5	2	65	120	6	6	0.29	0.12	140

[a]Using CPLEX 6.5 single processor version on a dual processor SUN Ultra 60 workstation
[b]Using ILOG Scheduler 4.4 and ILOG Solver 4.4 single processor versions on a dual processor SUN Ultra 60 workstation

MILP and CP to solve problems that are otherwise intractable if solved using either of the two methods alone. The class of problems considered in this paper has the characteristic that only a subset of the binary variables have a non zero objective function coefficients if modeled as an MILP. This class of problems can be formulated as a hybrid MILP/CP model that involves some of the MILP constraints, a reduced set of CP constraints, and the equivalence relations between the MILP and the CP variables. To solve this hybrid model, an MILP/CP based decomposition method and an LP/CP based branch-and-bound algorithm were proposed. Both of these algorithms rely on the same relaxed MILP and feasibility CP problems. The aim of these methods is to combine the strength of MILP for proving optimality by using the LP relaxations, and the power of CP for finding feasible solutions for hard discrete optimization problems by using the specialized propagation algorithms. It should be noted that the algorithms presented are fairly general and, if needed, alternative methods for solving feasibility problems may also be used.

As an application example, a scheduling problem was considered. The aim of this problem was to find a least-cost schedule to process a set of orders using dissimilar parallel machines subject to release and duedate constraints. It was shown that this problem can be modeled as an MILP, CP, or a combined MILP-CP OPL model (Van Hentenryck 1999). The computational results indicate that the CP model requires fewer constraints as compared to the MILP model. However, the computational effort required to solve the problems using any of the two models increases rapidly with problem size. The MILP-CP OPL model was larger in size than the CP model but was smaller compared to the MILP model. The combined OPL model could solve most of the problems faster relative to the MILP and CP models. The scheduling problem was then modeled as a hybrid MILP/CP model and solved using the proposed decomposition algorithm. Computational results indicate that for the larger instances of the example scheduling problem, the computational times were significantly smaller compared to MILP, CP, or combined MILP-CP OPL methods. The example problem demonstrates that the proposed methods can be computationally efficient for solving certain problems. It is acknowledged that even if a problem falls into the class of problems considered in this paper, the computational efficiency of the proposed algorithms depends strongly on the nature of the relaxed MILP problem, the feasibility CP problem, and the cuts that are used in the algorithm.

Acknowledgments

The authors would like to thank Greger Ottosson and Erlendur S. Thorsteinsson for interesting discussions about Constraint Programming and Iiro Harjunkoski for his help in revising the paper. Financial support from the National Science Foundation under Grant CTS-9810182 is gratefully acknowledged.

References

Applegate, D., B. Cook. 1991. A computational study of the job shop scheduling problem. *Operations Research Society of America* **3** 149–156.

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

Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsberg, P. H. Vance. 1998. Branch and price: column generation for huge integer programs. *Operations Research* **46** 316–329.

Bockmayr, A., T. Kasper. 1998. Branch-and-infer: a unifying framework for integer and finite domain constraint programming. *INFORMS J. Computing* **10** 287–300.

Carlier, J., E. Pinson. 1989. An algorithm for solving the job-shop problem. *Management Science* **35** 164–176.

Caseau, Y., F. Laburthe. 1994. Improving CLP scheduling with task intervals. P. Van Hentenryck, ed. *Logic Programming: Proceedings of the 11th International Conference*, MIT Press, Cambridge, MA. 369–383.

Caseau, Y., F. Laburthe. 1996. Cumulative scheduling with task intervals. M. Maher, ed. *Logic Programming: Proceedings of the 1996 Joint International Conference and Symposium*, MIT Press, Cambridge, MA. 363–377.

Chandru, V., J. Hooker. 1999. *Optimization Methods for Logical Inference*. John Wiley & Sons, Inc., New York.

Colmerauer, A. 1985. Prolog in 10 figures. *Communications of the ACM* **28** 1296–1310.

Darby-Dowman, K., J. Little. 1998. Properties of some combinatorial optimization problems and their effect on the performance of integer programming and constraint logic programming. *INFORMS J. Computing* **10** 276–286.

Darby-Dowman, K., J. Little, G. Mitra, M. Zaffalon. 1997. Constraint logic programming and integer programming approaches and their collaboration in solving an assignment scheduling problem. *Constraints, An International Journal* **1** 245–264.

Dincbas, M., P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf, F. Berthier. 1988. The constraint logic programming language CHIP. *FGCS-88: Proceedings of International Conference on Fifth Generation Computer Systems*, Tokyo, Japan. 693–702.

Geoffrion, A. M. 1972. Generalized Benders decomposition. *Journal of Optimization Theory and Applications* **10** 237–260.

Harjunkoski, I., V. Jain, I. E. Grossmann. 2000. Hybrid mixed-integer/constraint logic programming strategies for solving scheduling and combinatorial optimization problems. *Comp. Chem. Engng.* **24** 337–343.

Heipcke, S. 1999a. An example of integrating constraint programming and mathematical programming. *Electronic Notes in Discrete Mathematics* **1**.

Heipcke, S. 1999b. Comparing constraint programming and mathematical programming approaches to discrete optimisation-the change problem. *J. of Operational Research Society* **50** 581–595.

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

Hooker, J. N., M. A. Osorio. 1999. Mixed logic/linear programming. *Discrete Applied Mathematics* **96–97** 395–442.

Hooker, J. N., G. Ottosson, E. S. Thorsteinsson, H. J. Kim. 1999. On integrating constraint propagation and linear programming for combinatorial optimization. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, AAAI, The AAAI Press/MIT Press, Cambridge, MA. 136–141.

ILOG, Inc. 1999a. *CPLEX 6.5 User's Manual*. ILOG, Inc., Incline Village, NV.

ILOG, Inc. 1999b. *OPL Studio 2.1 User's Manual*. ILOG, Inc., Incline Village, NV.

ILOG, Inc. 1999c. *Scheduler 4.4 User's Manual*. ILOG, Inc., Incline Village, NV.

ILOG, Inc. 1999d. *Solver 4.4 User's Manual*. ILOG Inc., Incline Village, NV.

Le Pape, C. 1994. Implementation of resource constraints in ILOG schedule: a library for the development of constraint-based scheduling systems. *Intelligent Systems Engineering* **3** 55–66.

Marriott, K., P. J. Stuckey. 1998. *Programming with Constraints*. MIT Press, Cambridge, MA.

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

Ottosson, G., E. S. Thorsteinsson, J. N. Hooker. 2001. Mixed global constraints and inference in hybrid CLP–IP solvers. *Annals of Mathematics and Artificial Intelligence.* Forthcoming.

Padberg, M. W., G. Rinaldi. 1991. A branch-and-cut algorithm for a symmetric travelling salesman polytope. *SIAM Review* **33** 60–100.

Proll, L., B. Smith. 1998. Integer linear programming and constraint logic programming approaches to a template design problem. *INFORMS J. Computing* **10** 265–275.

Quesada, I., I. E. Grossmann. 1992. An LP/NLP based branch-and-bound algorithm for convex MINLP optimization problems. *Computers Chem. Engng.* **19** 937–947.

Raman, R., I. E. Grossmann. 1993. Symbolic integration of logic in MILP branch and bound techniques for the synthesis of process networks. *Annals of Operations Research* **42** 169–191.

Raman, R., I. E. Grossmann. 1994. Modelling and computational techniques for logic based integer programming. *Computers Chem. Engng.* **18** 563–578.

Rodosek, R., M. G. Wallace, M. T. Hajian. 1999. A new approach to integrating mixed integer programming and constraint logic programming. *Annals of Operational Research* **86** 63–87.

Smith, B. M., S. C. Brailsford, P. M. Hubbard, H. P. Williams. 1997. The progressive party problem: integer linear programming and constraint programming compared. *Constraints, An International Journal* **1** 119–138.

Tsang, E. P. K. 1993. *Foundations of Constraint Satisfaction*. Academic Press, Inc., San Diego, CA.

Van Hentenryck, P. 1989. *Constraint Satisfaction in Logic Programming*. MIT Press, Cambridge, MA.

Van Hentenryck, P. 1999. *The OPL Optimization Programming Language*. MIT Press, Cambridge, MA.

Wallace, M., S. Novello, J. Schimpf. 1997. ECLiPSe: a platform for constraint logic programming. *ICL Systems Journal* **12** 159–200.

Accepted by John N. Hooker; received October 1999; revised July 2000, Februray 2001; accepted March 2001.