Chapter 12

HYBRID MIP-CP TECHNIQUES TO SOLVE A MULTI-MACHINE ASSIGNMENT AND SCHEDULING PROBLEM IN XPRESS-CP

Alkis Vazacopoulos and Nitin Verma
Dash Optimization, Inc, 560 Sylvan Avenue, Englewood Cliffs, NJ 07632, USA

Abstract In this paper we introduce Xpress-CP–a Constraint Programming tool–and demonstrate its modeling and solving capabilities. We consider the multi-machine assignment and scheduling problem (Hooker et al. (1999)), where jobs, with release dates and deadlines, have to be processed on parallel unrelated machines (where processing times depend on machine assignment). Given a job/machine assignment cost matrix, the objective is to minimize the total cost while keeping all machine schedules feasible. We show that by deriving the benefits of MIP and CP techniques simultaneously this problem can be modeled and solved efficiently in a hybrid fashion using Xpress Optimization suite.

1. Introduction

Many real-world problems in supply chains, particularly scheduling, planning and configuration problems involving logic, constraint satisfaction, or discretization are difficult to solve using Mixed Integer Programming (MIP) techniques alone. Due to the large number of integer variables in the MIP formulation, the size of these problems may increase rapidly. Constraint Programming (CP) is an appropriate methodology for such combinatorial decision problems, since it provides an integrated framework for modeling such problems compactly, and supports user-controllable generalized solution procedures. However CP primarily targets constraint satisfaction. It lacks a global perspective, and is efficient only for small-term and medium-term planning problems. Hence, hybrid approaches using both CP and MIP methodologies provide promise for solving very large-scale problems while accounting for certain problem objectives. Such approaches are suitable for various job-shop scheduling, production and distribution planning, and vehicle scheduling and routing problems. The Xpress-MP suite facilitates such approaches. It consists of a framework for both CP and MIP technologies to exist together in Xpress' modeling language–Mosel, and provides tools to coherently exchange information between Xpress-Optimizer and Xpress-CP solver–CHIP. Other optimization packages such as ILOG Optimization Suite also provide similar frameworks. In this paper, we demonstrate the capabilities of Xpress-CP by using it for modeling and solving the Multi-Machine Assignment and Scheduling (MMAS) problem, originally proposed by Hooker et al. (1999). The structure of this problem makes it a suitable candidate for demonstrating Xpress-CP. Although the problem is a generalized and theoretical version of the problems occurring in job-shop, flow-shop and open-shop environments, its variations (see e.g., Pinedo and Chao (1998)) typically occur in production and scheduling problems in large scale supply chains.

1.1 Literature review and applications

The MIP based pure Branch and Bound (B&B) methodologies for solving scheduling problems suffer from the drawback of slow convergence due to weak LP relaxations and a large number of integer variables in the problem. Various specialized algorithms and heuristics have been proposed for single machine and parallel machine problems and their variations (by e.g., Brucker (2001), and Pinedo (1995)). Several constraint programming based techniques and search strategies have also been proposed (see e.g., Baptiste et al. (2001)) for problems based on the job-machine environment. The applicability of other hybrid approaches using CP in conjunction with MIP such as Branch and Infer (Bockmayr et al. (2003)), and Branch and Price (Easton et al. (2003)) have also been researched, and it has been demonstrated that they can be applied for solving supply chain planning and generalized assignment problems, respectively. Pure and hybrid CP applications have been presented in the scheduling sector in sports scheduling by Aggoun and Vazacopoulos (2004), in BASF's plastic production planning by Timpe[1], and integrated lot-sizing and scheduling of Barbot's paint production. Additionally, numerous applications from supply chains have been built using hybrid methods as mentioned by Bockmayr et al. (2003). The MMAS problem itself has been studied by Hooker et al. (1999), Jain and Grossmann (2001), Bockmayr et al. (2003), and recently by Sadykov and Wolsey (2003) in a hybrid optimization framework. Hooker et al. proposed a declarative hybrid modeling framework for the problem and a solution methodology which used CP to generate inequalities for the MIP component of the problem. Jain and Grossmann discussed a generalized class of problems where hybrid approaches can be used, and proposed an Iterative and a Branch and Cut (B&C) scheme for solving the problem. They demonstrated the iterative methodology for the MMAS problem by implementing it on a set of 10 problems. Bockmayr and Pisaruk implemented a B&C scheme on this problem that can be handled by CP using 'separation heuristics' for 'monotone constraints', and tested their implementation on a set of 8 MMAS problems that were minor variations of the 10 problems generated by Jain and Grossmann. Sadykov and Wolsey proposed some more variations of hybrid approaches using combinations of MIP, CP, Column Generation, and strong cuts (these cuts are a tighter version of the Preemptive cuts we propose). They implemented and compared the aforementioned combinations on 9 instances of MMAS problems taken from Bockmayr and Pisaruk, and on additional randomly generated data sets. In this paper, we discuss the architecture of Xpress-CP and demonstrate its capability using the MMAS problem. We propose certain Disjunctive and Preemptive cuts (see Section 3), and compare the results of the implementations of pure (MIP), Iterative (MIP-CP), and B&C (MIP-CP) methodologies. We also generate several random test cases and present the results of this implementation.

[1]Project supported by the European Commission, Growth Program, Research Project LISCOS -Large Scale Integrated Supply Chain Optimization Software, Contract G1RD-CT-1999-00034.

1.2 Outline

In Section 2 we identify the Assignment and the Scheduling components of the problem and present its pure MIP formulation. In Section 3, we discuss cuts for strengthening the LP relaxation of the problem. In Section 4, we review the methodologies proposed by Jain and Grossmann (2001), and Bockmayr et al. (2003) for solving the problem by combining the MIP and CP techniques. Then, in Section 5, we present the Xpress-CP implementation of the hybrid approaches, where we outline the Xpress-CP architecture, illustrate the Xpress-Mosel implementation, and provide our results. Finally in Section 6, we summarize our work and present conclusions.

2. The Multi-Machine Assignment and Scheduling Problem description

We consider a problem with N jobs and M parallel machines, where each machine can process one job at a time. Processing of the i^{th} job can start after the release time r_i and must end before the deadline d_i. The processing time and the cost of processing of i^{th} job on the m^{th} machine are p_{im} and c_{im}, respectively. Following common scheduling terminology, these machines are *unrelated* (Peter (2001)). The goal is to process each job on one of the machines in order to minimize the total processing cost. In the context of supply chains, the jobs can be viewed as various intermediate products that are supplied from an up-stream supplier on various days, and need to be further processed in one of the several factories before their delivery dates to down-stream buyers. The processing time then would be the total time to ship and process the products to factories, while the cost of processing would be the actual cost incurred in transportation and processing. Although all the factories might have the infrastructure for processing the products, the physical location, and the technology used in each factory may affect the total cost and times required to process the products. This problem can be viewed as an assignment and scheduling problem, where each job must be assigned to one of the available machines such that the total assignment cost is minimized, while maintaining the feasibility of the schedule of all jobs assigned to each of the machines (i.e., meeting the release-deadline restrictions of the jobs, while respecting their non-overlapping sequencing on the machines they are assigned to). In the next Section we present the standard MIP formulation of the assignment and the scheduling components of the problem.

2.1 MIP formulation

Sets:

$Jobs = \{1, \ldots, N\}$ (indexed by i, j, k).

$Machines = \{1, \ldots, M\}$(indexed by m, n).

Data:

r_i, d_i: Release and deadline times, respectively.

p_{im}, c_{im}: Processing time and cost of processing, respectively.

Variables:

x_{im}: Binary variable equal to on if job i is assigned to machine m, and 0 otherwise.

s_i: Starting time of processing of job i.

y_{ij}: Binary variable equal to one if job i precedes job $j \neq i$ when both of them are assigned to the same machine, and 0 otherwise.

Basic Formulation:

Assignment

$$\begin{aligned}
\min \quad & \sum_{i \in Jobs, m \in Machines} c_{im} x_{im} \\
s.t. \quad & \sum_{m \in Machines} x_{im} = 1, \ \forall i \in Jobs, \\
& x_{im} \in \{0, 1\}, \ \forall i \in Jobs, \ m \in Machines.
\end{aligned}$$

Scheduling

$$
\begin{aligned}
& r_i \le s_i \le d_i + \sum_{m \in Machines} p_{im} x_{im}, \ \forall i \in Jobs, \\
& x_{im} + x_{jm} - 1 \le y_{ij} + y_{ji}, \\
& \qquad\qquad \forall i, j \in Jobs: \ j > i, m \in Machines. \\
& s_i + \sum_{m \in Machines} p_{im} x_{im} \le \\
& s_j + \left(\sum_{k \in Jobs} \max_{m \in Machines} \{p_{km}\} \right) (1 - y_{ij}), \ \forall i, j \in Jobs : i \ne j, \\
& s_i \ge 0 \ \forall i \in Jobs, \ y_{ij} \in \{0, 1\}, \ \forall i, j \in Jobs: \ i \ne j.
\end{aligned}
$$

The above formulation for the "assignment" portion of the problem is straightforward. The scheduling part is slightly more complex. The first scheduling constraint ensures that the job is scheduled within its time window. The second equation states that when two jobs i and j are assigned to the same machine m, then either i precedes j or j precedes i. The third equation links starting time variables to disjunction variables as follows: Given two jobs i and j, if i precedes j then, between the starting time of i and the starting time of j, there must be enough time to schedule job i, i.e.,

$$s_i + \sum_{m \in Machines} p_{im} x_{im} \le s_j$$

Now, if i does not precede j, the third equation leads to

$$s_i + \sum_{m \in Machines} p_{im} x_{im} \le s_j + \left(\sum_{k \in Jobs} \max_{m \in Machines} \{p_{km}\} \right)$$

which always holds since the right term is a large constant value (i.e., a "big M"). Such values are known to be a major issue for MIP formulations because they lead to very poor relaxations. An alternative for the scheduling part would be to use a time indexed formulation (see for instance Pritsker et al. (1969)) in which a binary variable is associated with each candidate starting time for each activity. As noticed by several researchers, this leads to huge MIP problems that are not likely to be solved, even for medium sized instances.

3. Cuts to tighten the MIP Formulation

The solution time to solve the problem can be improved significantly by addition of the following cuts:

Maximum Duration Cuts

$$\sum_{i \in Jobs} p_{im} x_{im} \leq \max_{i \in Jobs} \{d_i\} - \max_{i \in Jobs} \{r_i\}, \ \forall m \in Machines.$$

The above cuts–originally proposed by Jain and Grossmann (2001)–ensure that the total processing time of the jobs assigned to a machine should not exceed the maximum possible time $(\max_{i \in Jobs} \{d_i\} - \min_{i \in Jobs} \{r_i\})$ available.

Disjunctive Cuts

If the total processing time of a pair of jobs on a machine exceeds the maximum span of time available for processing, then both of them can not be assigned to the machine; hence, we propose the following Disjunctive cuts.

$$x_{im} + x_{jm} \leq 1, \qquad \forall i, j \in Jobs : j > i$$
$$m \in Machines : p_{im} + p_{jm} > \max(d_i - r_j, d_j - r_i).$$

Preemptive Cuts

$$\sum_{\substack{k \in Jobs \\ r_i \leq r_k, d_k \leq d_j}} p_{km} x_{km} \leq d_j - r_i, \quad \forall i, j \in Jobs : d_j > r_i, m \in Machines :$$
$$\sum_{\substack{k \in Jobs \\ r_i \leq r_k, d_k \leq d_j}} p_{km} > d_j - r_i.$$

In the context of MMAS problem we also propose the Preemptive cuts which are similar to the Disjunctive cuts. They impose the restriction that the total processing time of the set of jobs that are released after the release time r_i, are due before d_j, and are assigned to a machine, should not exceed the time-span d_j - r_i. Surprisingly, these cuts are very powerful. They ensure that, once the assignment problem is solved, we have a feasible preemptive schedule on each machine (Carlier and Pinson (1990)). In a recent paper by Sadykov and Wolsey (2003), the authors have also proposed methods which further strengthen the Preemptive cuts, and implemented them in a hybrid fashion using MIP and CP (see MIP$^+$/CP in Sadykov and Wolsey (2003)).

Logical Cuts for the pure MILP formulation

The following set of cuts Jain and Grossmann (2001) can be used in the pure MILP formulation (see Section 2.1).

$$\begin{aligned} y_{ij} + y_{ji} &\le 1, && \forall i,j \in Jobs : j > i, \\ y_{ij} + y_{ji} + x_{im} + x_{jn} &\le 2, && \forall i,j \in Jobs : j > i, \\ & && m,n \in Machines : m \neq n. \end{aligned}$$

The first set of cuts ensures that given a pair of jobs assigned to a machine, one of them should precede the other. The second set of cuts prevents the sequencing of a pair of jobs on any machine if they are assigned to different machines. Alternatively, if they are sequenced on a machine, they cannot be assigned to different machines.

4. Hybrid approaches using Constraint Programming

The introduction of binary variables for sequencing the jobs (y_{ij}) on the machines, together with the associated constraints, makes this problem difficult to solve using MIP techniques alone. Since the scheduling component of the problem does not affect the objective function, one can solve this problem faster by applying a hybrid scheme using both MIP and CP. MIP can be used for solving the assignment problem as a master problem, and CP for checking the feasibility of the scheduling problem as a sub-problem. Whenever the sub-problem is infeasible, a cut is added to the master problem. An iterative scheme implemented by Jain and Grossmann (2001) is outlined in Section 4.2, and a B&C scheme implemented by Bockmayr et al. (2003) is discussed in Section 4.3.

4.1 CP scheduling formulation

CP can be used for checking the feasibility of scheduling the processing of jobs on the machine to which they are assigned. Let be the current solution vector to the master problem. Then define $Jobs^m = \{i \in Jobs : x_{im} = 1, x_{jm} \in \{0,1\}\ \forall j \in Jobs\}$ as the set of jobs assigned to machine m based on the current solution.

The CP problem $\forall m \in Machines : Jobs^m \neq \emptyset$ is stated as -

$$
\begin{aligned}
&i.start \in [r_i, d_i - p_{im}]\ \forall i \in Jobs^m\\
&i.duration = p_{im}\ \forall i \in Jobs^m\\
&disjunctive(Jobs^m)
\end{aligned}
$$

The first set of constraints sets the domains of start times of the assigned jobs, while the second set fixes their processing durations on the machine. The disjunction ensures that the sequencing of the jobs is non-overlapping. If the CP problem is infeasible then a "**no good**" cut (Hooker et al. (1999)): $\sum_{i \in Jobs^m} x_{im} \leq x_{im} \leq |Jobs^m| - 1$ may be added to the master problem.

Very efficient constraint propagation techniques, known as "edge-finding", have been developed to solve such scheduling problems (see Carlier and Pinson (1990); Baptiste et al. (2001)). Edge-finding bounding techniques are particular constraint propagation techniques which reason about the order in which several jobs can be processed on a given machine. It consists of determining whether a job can, cannot, or must execute before (or after) a set of jobs which require the same machine. Two types of conclusions can then be drawn: new ordering relations ("edges" in the graph representing the possible orderings of activities) and new time-bounds, i.e., strengthened earliest and latest start and end times of activities.

4.2 Iterative method

In the iterative method, the assignment problem is solved repetitively as an MIP master problem and the "no good" cuts generated from the CP sub-problem are added to it. The loop terminates when the master problem is found to be infeasible or when the MMAS problem is optimal.

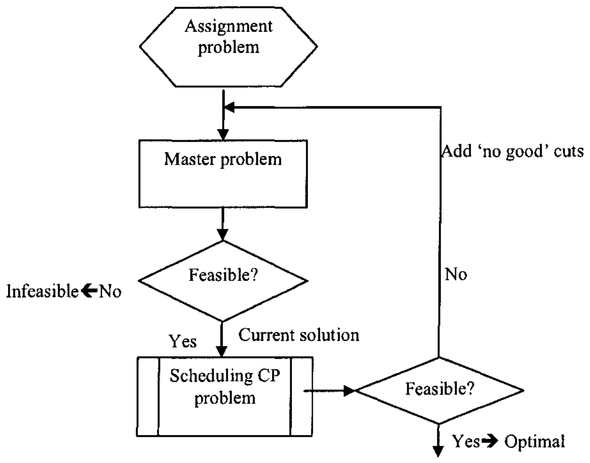

Figure 12.1. Iterative method

4.3 Branch and Cut (B&C) method

The B&C method is similar to the iterative method except that in this method, the master problem is not solved to optimality before adding cuts. Instead, the assignment problem is solved as an MIP problem using the standard B&B method. At each partially feasible node (where one or more machines have been assigned jobs), CP is used to generate the "no good" cuts if possible, thereby ensuring that each integer feasible node is also feasible for the MMAS problem.

5. Implementation in Xpress-CP

The normal approach to solve problems using such a hybrid procedure is to have two models–a planning model and a scheduling model–with the former solved with MIP and the latter solved with CP. The need to use two models and two separate systems increases the complexity, reliability, and lifecycle cost of the system. It also requires some manual intervention to iterate between the two systems, which is expensive and unreliable. Xpress-CP overcomes this limitation by providing a unified framework for modeling problems in a single model, as discussed next.

5.1 Xpress-CP design and capabilities

Xpress-CP combines the mathematical optimization software Xpress-MP and the constraint programming software CHIP in a hybrid optimization framework. An important advantage of Xpress-CP is that the MIP and CP technologies exist within the same software environment and the problem is expressed within a single model. The model is written in Xpress-Mosel and the MIP/CP solver is invoked without the need for complex programming or particular expertise in MIP or CP. Built on Mosel's *Native Interface* technology (Colombani and Heipcke (2002)), Xpress-CP provides a rich collection of *types* and constraint structures to represent problems in supply chain optimization for advanced planning and scheduling. The Mosel module 'xpresscp' provides a high level abstraction by *various types* such as cpdvar, cpoperation, cpnoonoverlap, cplabeling, etc; and supports several functions and *procedures*. Specifically, Xpress-CP provides:

- high level semantic objects such as operations and machines which are part of the language used by the end users to describe problems;
- functionality to get and set the attributes of these objects;
- high level semantic constraints which ease the modelling of the problem's constraints;
- high level primitives to guide during the search procedure;
- various predefined strategies and heuristics.

Various *types*, *procedures* and *functions* defined in Xpress-CP, together with Xpress-CP's superior design enable easy and concise modeling of complex scheduling problems in Mosel

5.2 Mosel model

The Xpress-Optimizer module 'mmxprs' and the Xpress-CP module 'xpresscp' are used for writing the iterative (Section 4.2) and the B&C (Section 4.3) schemes in Xpress-Mosel. In the following section only the relevant parts of the Mosel model are described. The complete model with data files can be obtained from the authors upon request.

First the model entities defined in Section 2.1 are declared as follows:

```
declarations
  Jobs=1,..,N
  Machines=1,..,M
  r,d:array(Jobs) of integer
  p,c:array(Jobs,Machines) of integer
  x:array(Jobs,Machines) of mpvar
end-declarations
```

5.2.1 Assignment. Next the assignment problem is written as follows in Mosel:

```
TotalCost := sum(i in Jobs,m in Machines) c(i,m)*x(i,m)
forall(i in Jobs) Assignment(i) := sum(m in Machines) x(i,m)=1
forall(i in Jobs,m in Machines) x(i,m) is_binary
```

5.2.2 Cuts. The Maximum Duration, Disjunctive and Preemptive cuts are written as follows in Mosel:

```
MaxDuration:=max(i in Jobs) d(i)-min(i in Jobs) r(i)
forall(m in Machines)
  MaximumDuration(m):=sum(i in Jobs) p(i,m)*x(i,m)¡=MaxDuration

forall(i,j in Jobs | i>j)
  forall(m in Machines |
p(i,m)+p(j,m)>maxlist(d(i)-r(j),d(j)-r(i)))
    Disjunctive(i,j,m):=x(i,m)+x(j,m)<=1

forall(i,j in Jobs | r(i) <= r(j) and d(i) <= d(j)) do
  S:=union (k in Jobs | r(i)<=r(k) and d(k)<=d(j)) {k}
  forall(m in Machines | sum(k in S) p(k,m)>(d(j)-r(i)))
    Preemptive(i,j,m):=sum(k in S) p(k,m)*x(k,m)<=d(j)-r(i)
end-do
```

5.2.3 CP formulation for feasibility check. The CP formulation of the problem discussed in Section 4.1 is done by declaring the operations for each job, followed by setting the domains of their starting and processing times.

```
declarations
  o:array(Jobs) of cpoperation
```

```
end-declarations

forall(i in Jobs) do
  max_dur:=max(m in Machines) p(i,m)
  min_dur:=min(m in Machines) p(i,m)
  cpsetstart(o(i),r(i),d(i)-min_dur)
  cpsetduration(o(i),min_dur,max_dur)
end-do
```

The function cpsetstart() sets the domain for the starting times, and the function cpsetduration() sets the bounds on processing times. Now, given a machine m and a set of jobs assigned to it, CP can be used to check the feasibility with respect to scheduling and sequencing of the jobs as follows:

```
function IsFeasible(m:integer,JobsAssigned:set of integer):boolean
  cpstart
  forall(i in JobsAssigned) do
    fes_asgn:=
      if ((r(i)+p(i,m))>d(i),false,
        cpsetminmax(cpgetstart(o(i)),r(i),d(i)-p(i,m)))
    fes_asgn:=
      if (fes_asgn,
        cpsetminmax(cpgetduration(o(i)),p(i,m)),false)
  end-do
  fes_asgn:=if (fes_asgn,CheckFeasibility(JobsAssigned),false)
  cpend
  returned:=fes_asgn
end-function
```

Here, Xpress-CP is invoked by cpstart(), and terminated by cpend(). It is ensured that all the jobs under consideration can be assigned to the machine m simply based on their processing times on the machine and the spans of time available between their release and deadline times. Then, the function cpsetminmax() is used to set the domains of starting and processing times based on the current assignment. The feasibility of current assignment is checked by calling a user-defined function Check-Feasibility() as follows:

```
function CheckFeasibility(JobsAssigned:set of integer):boolean
  declarations
    Oprtns:set of cpoperation
```

```
        Disjunctive:cpnonoverlap
        Label:cplabeling
      end-declarations

      returned:=false
      forall(i in JobsAssigned) Oprtns += {o(i)}
      cpsetops(Disjunctive,Oprtns)
      if cppost(Disjunctive) then
        cpsetops(Label,Oprtns)
        cpsetselectop(Label,"start",1,"smallest")
        cpsetseelectval(Label,"start",1,"indomain_min")
        returned:=cppost(label)
      end-if
    end-function
```

In the above functions, all the operations are collected in a set Oprtns, and assigned to a non-overlap type constraint Disjunctive. Next, the feasibility of the current assignment is checked by posting the disjunctive constraint to the CP-solver. The search strategy for finding a solution is defined by creating a primitive Label for the operations, and setting the operation-selection and value-selection criterions. In the MMAS problem we define a high priority search strategy by selecting the smallest values of starting times of operations from their domains. The function CheckFeasibility() returns false if the assigned jobs cannot be sequenced on machines.

5.2.4 Iterative schematics. The Mosel implementation of the iterative scheme discussed in Section 4.2 is as follows:

```
minimize(TotalCost)
while (true) do
  if getprobstat<>XPRS_OPT then break;end-if
  savebasis("previous optimal basis")
  ncut:=0
  forall(m in Machines) do
    JobsAssigned:=union(i in Jobs| getsol(x(i,m))=1) {i}
    NumJobs:=getsize(JobsAssigned)
    if NumJobs>0 then
      if not IsFeasible(m,JobsAssigned,false) then
        TotNumOfCuts+=1
        ncut+1=1
        cuts(TotNumOfCuts):=
            sum(i in JobsAssigned) x(i,m)<=(NumJobs-1)
          end-if
        end-if
      end-do
      if ncut>0 then
        loadprob(TotalCost)
        loadbasis("previous optimal basis")
        minimize(TotalCost)
      else
        break
      end-if
    end-do
```

The master problem is solved repetitively in the loop which terminates when no more cuts can be added to it or when it is infeasible or unbounded. The feasibility of the sub-problem is checked using CP by calling the function IsFeasible() shown in Section 5.2.3. The global counter TotNumCuts (declaration not shown here) is used to keep track of number of cuts added to the master problem. The 'mmxprs' functions savebasis() and loadbasis() are used to save and load the optimal basis respectively during each iteration which helps in 'warm starting'.

5.2.5 B&C call-back schematics. The logic behind solving the problem using the B&C scheme is similar to that of the iterative *scheme*, except that now the cuts are added during the B&B search. This is achieved in Mosel by turning off the Xpress pre-solver so that the matrix structure is not lost, and directing the Xpress cut-manager to call a user-defined routine at every node of the B&B tree.

The optimizer is intercepted during the B&B search by setting a call-back from the cut-manager as follows:

```
setcallback(XPRS_CB_CUTMGR,"EveryNode")
```

where the function EveryNode() is defined as follows:

```
function EveryNode: boolean
  returned:=false
  setparam("xprs_solutionfile",0)
    forall(i in Jobs,m in Machines) CurrSol(i,m):=getsol(x(i,m))
  setparam("xprs_solutionfile",1)
  loadcuts(NO_GOOD,1)
```

```
    ncut:=0
    forall(m in Machines) do
      if and(i in Jobs) (CurrSol(i,m)=0 or CurrSol(i,m)=1) then
        JobsAssigned:=union(i in Jobs| CurrSol(i,m)=1) {i}
        NumJobs:=getsize(JobsAssigned)
        if NumJobs>0 then
          if not IsFeasible(m,JobsAssigned,false) then
            TotNumOfCuts+=1
            ncut+=1
            cut:=sum(i in JobsAssigned) x(i,m)-(NumJobs-1)
            addcut(NO_GOOD,CT_LEQ,cut)
            returned:=true
          end-if
        end-if
      end-if
    end-do
  end-function
```

The routine EveryNode() is called at each node after the LP relaxed problem at that node is solved. The solution at the current node is stored in CurrSol (declaration not shown here) by turning off Xpress' solutionfile temporarily, so that the solution is read from the Optimizer directly instead of the solution file which stores the current best solution. The "no good" cuts are added to the matrix at current node by calling the 'mmxprs' function addcut(). Since these cuts are globally valid, they are loaded by calling the 'mmxprs' function loadcuts(). The constant NO_GOOD (declaration not shown here) is used as an identifier for the cuts.

5.3 Computational Results

In the following sections we present the results of implementation in Xpress-CP. The experiments were done on a P-IV, 2.2GHz machine with 1GB RAM. The default time limit for running the model was set to 1 hour. The Xpress pre-solver is turned off for the B&C method. We used the Maximum-duration, Disjunctive and Preemptive cuts in our implementation. The Xpress components and their version number, used for implementing the hybrid schemes were:

- Mosel (modeling language)- 1.2.4
- IVE (Integrated Visual Environment)- 1.14.70

- mmxprs (Mosel module for MIP)- 1.2.3
- xpresscp (Mosel module for CP)- 0.1.5
- Optimizer (Xpress LP/MIP/QP/MIQP solver)- 14.27
- CHIP (CP solver)- 5.5

5.3.1 Comparison with Jain and Grossmann's results. Table 5.3.1 shows the comparison with results of Jain and Grossmann's implementation. They used CPLEX 6.5 single processor version on a dual processor SUN Ultra 60 workstation. The problems were originally generated by Jain and Grossmann.

The above table lists the number of major iterations in the Iterative method, the number of cuts added in the B&C approach, the number of nodes in the B&B tree, and corresponding times in seconds to achieve an optimal solution. Since the times required for solving these problems are quite small and the platform used by Jain and Grossmann is different than ours, it is hard to compare the results. As far as the iterative and the B&C methods in Xpress-CP are concerned, the latter seems to solve the problem faster than the former.

5.3.2 Comparison with Bockmayr and Pisaruk's and Sadykov and Wolsey's results. The following table shows a comparison with results of Bockmayr and Pisaruk's, and Sadykov and Wolsey's implementations. The tested problems are essentially variations of the problems generated by Jain and Grossmann. Bockmayr and Pisaruk used a similar B&C method together with a heuristic for binary variables and a set of cycle cuts, and implemented it using Xpress-MP 2003B, CHIP version 5.3 on a Pentium III machine with 600 MHz with 256 MB memory. Sadykov and Wolsey carried out all the experiments on a PC with P-IV 2 GHz processor and 512 MB RAM. They also used MIP and CP in B&C, together with a tighter version of the Preemptive cuts (MIP$^+$/CP) and implemented it in Xpress-Mosel 1.3.2, Xpress-Optimizer 14.21 and CHIP 5.4.3.

The above table also lists the best objective value and the best bound for the problems that were not solved to optimality (The corresponding entries for the problem solve to optimality are marked by *). Given their machine specifications, it is observed from Table 5.3.2 that Bockmayr

Problem	obj	Jain & Grossmann: Iterative			Xpress-CP: Iterative			Xpress-CP: B&C		
	val	#iteration	#cuts	time(s)	#iterations	#cuts	time(s)	#cuts	#nodes	time(s)
1a	26	2	1	0.02	1	0	0.11	0	1	0.079
1b	18	1	0	0.01	1	0	0.094	0	1	0.062
2a	60	13	16	0.52	1	0	0.188	1	10	0.219
2b	44	1	0	0.02	1	0	0.109	0	1	0.093
3a	101	31	43	4.18	2	1	0.735	7	219	0.5
3b	83	1	0	0.02	1	2	0.11	0	1	0.078
4a	115	18	26	2.25	3	0	1.172	4	174	0.66
4b	102	1	0	0.04	1	0	0.109	0	1	0.093
5a	158	31	60	14.13	5	6	7.052	11	557	2.344
5b	140	6	6	0.41	1	0	0.156	0	1	0.125

Table 12.1. Comparison with Jain and Grossmann's results.

Problem	obj	Bockmayr & Pisaruk: B&C			Sadykov & Wolsey: MIP$^+$/CP	Xpress-CP: B&C			
	obj	#cuts	#nodes	time(s)	time(s)	#cuts	#nodes	time(s)	bestobj/bestbound
3_12a	101	93	78	0.47	0.407	7	219	0.5	*
3_12b	104	210	149	0.87	0.328	0	640	1.16	*
5_15a	115	104	144	0.7	0.563	4	174	0.66	*
5_15b	137	164	177	0.9	0.563	3	155	0.86	*
5_20a	159	360	924	4.01	1.313	11	557	2.34	*
5_20b	145	3179	38303	245.77	2.703	136	46338	150.51	*
6_24	231	9901	389372	5762.89	3.672	24	1800	8.31	*
7_30	213	-	-	>10hr	16.656	67	234351	3848.73	214/211.378
8_34[2]	252	NA	NA	NA	3916.42	564	106123	3828.39	255/247.124

Table 12.2. Comparison with Bockmayr and Pisaruk's results.

and Pisaruk's results might be better than ours. This can be attributed to the fact that they use a specialized heuristic and cycle cuts on top of the B&C method which seems to enhance the performance. Similarly, tightening of the Preemptive cuts by Sadykov and Wolsey significantly improves the performance.

5.3.3 Comparison with Randomly generated data.

We generate random data sets in Mosel as follows:

$$c_i \in \{1, \ldots, 20\} \ \forall i \in Jobs$$
$$r_i \in \{1, \ldots, 20\} \ \forall i \in Jobs$$
$$d_i \in \{15, \ldots, 25\} \ \forall i \in Jobs$$
$$p_{im} \in \{1, \ldots, d_i - r_i - 1\} \ \forall i \in Jobs, m \in Machines$$

The problems are generated by varying the number of machines M, the number of jobs N, and the seed for generating the random data. The results of the pure MILP (Xpress pre-solver is on), Iterative (Xpress pre-solver is off), and B&C implementations are tabulated in the following Table. Note that only the non-trivial problems (that require one or more cuts by either of the hybrid methods) are shown below.

From the above results it is observed that the times required for solving the problems using hybrid-schemes are much less than when using pure MILP. Additionally, the B&C method solves most of the problems faster than the Iterative method.

M	N	seed	obj	MILP			Iterative			B&C		
			val	bestobj/bestbound	#nodes	time(s)	#cut	#iterations	time(s)	#cut	#nodes	time(s)
	15	1	95	*	98	5.516	1	2	0.657	4	22	0.281
	20	1	145	*	3352	105.34	3	4	14.485	15	755	4.328
		2	135	*	1170	67.015	0	1	4.797	7	382	2.469
5	25	1	192	*	3297	255.47	0	1	6.078	12	1051	7.812
		2	234	*	6425	219.52	0	1	5.954	12	570	5.547
	30	1	247	inf/240	104523	3653.8	5	4	47.296	19	1659	19.016
		2	192	*	29379	1633.6	0	1	12.437	7	494	7.797
	25	1	86	*	39	27.109	1	2	0.531	1	1	0.234
		2	84	*	4154	359.59	3	4	6.015	5	23	2.203
10	30	2	100	*	11596	2310.9	7	4	19.625	12	199	4.563
	40	1	171	214/167	1976	3614.8	5	5	355.64	43	4214	201.25
		2	145	195/142	2635	3618.5	1	2	68.672	7	190	28.687
	40	1	86	*	1144	2134.6	4	3	23.25	4	11	6.578
	50	1	111	inf/110	173	3600	6	5	261.09	7	17	76.218
	60	1	137	inf/134.3	87	3657.8	5	4	640.59	20	300	248.14
20		2	131	165/131	188	3652.6	4	3	108.77	4	13	79.64
	70	1	171	inf/168.558	30	3932.41^c	8	5	3154.1	10	219	1108.7
		2	178	inf/177.309	21	3693.1	2	2	1197.1	10	203	886.83

Table 12.3. Results for the randomly generated problems.

6. Summary and Conclusion

In this paper we demonstrated the capabilities of Xpress-CP, which provides a natural syntax for expressing scheduling problems, and presented the Xpress-Mosel framework which facilitates the existence of both MIP and CP technologies to co-exist and enable rapid modeling and solving. We considered the Multi-machine assignment and scheduling problem, which, because of its structure, is a perfect candidate for demonstration purposes. We began by presenting the pure MIP formulation of the problem and cuts that could be used for strengthening its linear relaxation. Next, we showed two hybrid approaches to solve the problem, namely Iterative, and B&C, followed by illustrating the implementation of these approaches in Mosel. Finally, we compared our results with those of Jain and Grossmann's, Bockmayr and Pisaruk's, and Sadykov and Wolsey's. We also compared the Iterative and the B&C methods for various problems generated in Mosel randomly. From the results it was observed that the B&C approach solves the problem much faster than the iterative approach in most of the cases, and using stronger cuts further improves the performance significantly.

Acknowledgments

The authors would like to thank Philippe Baptiste for many enlightening discussions on the cuts mentioned in Section 3 for the MMAS problem, and for his help in revising the paper.

References

Aggoun, A. and Vazacopoulos, A. 2004. Solving Sports Scheduling and Time tabling Problems with Constraint Programming, in Economics, Management and Optimization in Sports, Edited by S. Butenko, J. Gil-Lafuente and P.M. Pardalos, Springer.

Baptiste, P., Le Pape, C. and Nuijten, W. 2001. Constraint Based Scheduling. Kluwer.

Bockmayr, A. and Kasper, T. 2003. Branch-and-infer: A framework for combining CP and IP. In Constraint and Integer Programming (Ed. M. Milano), Chapter 3, 59 - 87, Kluwer.

Bockmayr, A. and Hooker, J.N. 2003. Constraint programming. In Handbooks in Operations Research and Management Science: Discrete Optimization (Eds. K. Aardal, G. Nemhauser, and R. Weismantel), Elsevier, To appear.

Bockmayr, A. and Pisaruk, N. 2003. Detecting Infeasibility and Generating Cuts for MIP using CP. 5th International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR'03, Montreal, May 2003.

Brucker, P. 2001. Scheduling Algorithms. Third Edition, Springer.

Carlier, J. and Pinson., E. 1990. A Practical Use of Jackson's Preemptive Schedule for Solving the Job-Shop Problem. Annals of Operations Research 26, 269-287.

Colombani, Y and Heipcke, S. 2002. Mosel: An Overview. May 2002, available at http://www.dashoptimization.com/home/downloads/pdf /mosel.pdf.

Easton, K., Nemhauser, G. and Trick, M. 2003. CP Based Branch-and-Price. In Constraint and Integer Programming (Ed. M. Milano), Chapter 7, 207 - 231, Kluwer.

Hooker, J.N., Ottosson, G., Thorsteinsson, E.S. and Kim, H.J. 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.

Jain, V. and Grossmann, I.E. 2001. Algorithms for hybrid MIP/CP models for a class of optimization problems. INFORMS J. Computing, 13(4), 258-276, 2001.

Peter, B. 2001. Scheduling Algorithms. Springer Lehrbuch.

Pritsker, A., Watters, L. and Wolfe, P. 1969. Multi-project scheduling with limited resources: a zero-one programming approach. Management Science, 16:93-108.

Pinedo, M. 1995. Scheduling: Theory, Algorithms and Systems. Prentice - Hall, NJ.

Pinedo, M. and Chao, X. 1998. Operations Scheduling with Applications in Manufacturing and services. McGraw-Hill/Irwin.

Sadykov, R. and Wolsey, L. 2003. Integer programming and constraint programming in solving a multi-machine assignment scheduling problem with deadlines and release dates. CORE discussion paper, Nov 2003.