Logic-based Multi-Objective Optimization for Restoration Planning

Jing Gong[1], Earl E. Lee[2], John E. Mitchell[3], William A. Wallace[4]

[1] DSES, Rensselaer Polytechnic Institute, Troy, NY 12180. gongj@rpi.edu
[2] DSES, Rensselaer Polytechnic Institute, Troy, NY 12180. leee@rpi.edu
[3] Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180. mitchj@rpi.edu
[4] DSES, Rensselaer Polytechnic Institute, Troy, NY 12180. wallaw@rpi.edu

Summary. After a disruption in an interconnected set of systems, it is necessary to restore service. This requires the determination of the tasks that need to be undertaken to restore service, and then scheduling those tasks using the available resources. This paper discusses combining mathematical programming and constraint programming into multiple objective restoration planning in order to schedule the tasks that need to be performed. There are three classical objectives involved in scheduling problems: the cost, the tardiness, and the makespan. Efficient solutions for the multiple objective function problem are determined using convex combinations of the classical objectives. For each combination, a mixed integer program is solved using a Benders decomposition approach. The Master Problem assigns tasks to workgroups, and then subproblems schedule the tasks assigned to each workgroup. Hooker has proposed using integer programming to solve the master problem and constraint programming to solve the subproblems, when using one of the classical objective functions. We show that this approach can be successfully generalized to the multiple objective problem. The speed at which a useful set of points on the efficient frontier can be determined should allow the integration of the determination of the tasks to be performed with the evaluation of the various costs of performing those tasks.

Key words: Constraint programming, Mixed integer programming, Multi-Objective, Scheduling and planning.

1 Problem Description

Our previous work [1], [2] introduced the interdependent layered network model (ILN). This model was a network flow based model of civil infrastructure systems incorporating their interdependencies (or interconnectedness). The work identified five types of interdependency and mathematical representations of each were developed. This model of the system of systems could demonstrate the cascading effects of a disruption; allow for collaborative

restoration planning across the set of systems; and could show vulnerability in a system due to its reliance on other systems.

In [1], a scenario was developed to exercise the model. Using data provided by the respective system managers, a realistic representation of the power, communications and subway systems of a large portion of Manhattan was developed. A disruption with effects similar to the September 11, 2001 attacks on the World Trade Center was proposed. The disruption was entered into the ILN with the output showing the service outages which resulted directly from the disruption and the impact due to the cascading effect of the disruption due to the interconnectedness of the systems.

The next step was the development of a restoration plan which met the constraints placed upon planners by the various management agencies involved. In general, the restoration plan consisted of the running of temporary lines along the streets of Manhattan to restore power and phone service until permanent repairs could be made. New constraints included the capacity of the temporary lines, limits on streets where lines may or may not be run along or crossed, etc. Consider the street sections in the area of interest as arcs. The output of the ILN was a subset of those arcs which met the constraints of the planners and restored the services. With a plan developed, another module of the ILN developed a schedule for the set of tasks. This scheduling module, like the ILN was a mixed-integer problem.

This paper builds upon that work. In this case, a hybrid mixed integer and constraint programming modeling framework is presented for scheduling. Based on the optimal restoration plan developed in the ILN, this hybrid model determines how to accomplish the plan, i.e. the assignment and sequence of activities.

In the example provided in this paper, all resources are considered unary. Workers and equipment are bundled into work groups and have sufficient skills to accomplish any of the tasks in the set. Each task only requires one workgroup. The only differences between the work groups are their cost and time required to complete each task. So the decision problem is how to assign repair tasks to these groups and schedule tasks for each workgroup in an optimal fashion. Future work will include shared resources. For example, supervisors might be a shared resource. Each task would have its own unique requirement for the number of supervisors at each job site, with a fixed number of supervisors available during each time interval.

Requirements from a planner or a manager could be: spend as little money as possible; complete all tasks by their due dates; or finish all tasks as soon as possible. So there are three measures that need to be minimized: the cost, the tardiness of each task and the makespan of all tasks. They are formulated as objectives in our model. Our approach to address this multi-objective issue is to minimize the weighted sum of the three objectives. Decision making for the optimal solution can be viewed as a procedure to tradeoff among them.

2 Literature Review

Constraint Programming (CP) developed as a computer science technology which employs developments in artificial intelligence and computer programming languages [3]. It provides the capability of defining the structure of the search space and specifying the algorithm for exploring the search space, which make it possible to solve some particular problems efficiently, e.g. some combinatorial optimization problems [4]. Constraint programming is viewed as more like a method of solving the problem, not just a modeling language, like AMPL, GAMS and so on, although languages supporting constraint programming have a strengthened expressiveness compared with those traditional mathematical programming languages. More and more researchers discuss incorporation of this technology into the operations research field.

Recently, research interest in combining constraint programming with mathematical programming arose. Kim and Hooker [5] applied a hybrid solution method which combines constraint programming and linear programming to fixed charge network flow problems. They solved the problem by combining constraint propagation and a projected relaxation and got a significant computational performance improvement compared with a commercial mixed integer programming code. Hooker [6] compared those two technologies and pointed out that they have complementary strengths in solving integer and mixed integer problems, although one originated from mathematics and the other from engineering. Hooker [7] proposed a search-infer-relax framework for integrating solution methods. Searching is a procedure of enumerating all values in the domain. Inference derives implicit constraints to reduce the domain. Relaxation solves a relaxed problem for a bound on the original problem. Branch-and-bound in mathematical programming, specifically, integer programming and mixed integer programming, is a particular strategy of searching the whole space. It can be viewed as a sophisticated enumeration, which makes it possible to combine two solution methods in this integration framework.

Another integration scheme is applying constraint programming into classical Benders decomposition for some problems whose subproblem is easily solved by constraint programming technology. Hooker [8] extended the idea of the classical Benders decomposition to a logic-based Benders. The logic-based Benders cuts are obtained by solving the inference dual of the subproblem. The solution of the inference dual can prove optimality when variables of the master problem take certain values. The difference between the logic-based Benders cut and classical Benders cut is that no standard form exists for the logic-based cut. The subproblem could be a Linear Program (LP), Mixed Integer Program (MIP), or CP, and cuts are generated by logical inference on the solution of the subproblem. However, generally the master problem is an MIP, therefore cuts are formulated as linear constraints. The classical Benders decomposition is strengthened by introducing the logic-based Benders cut.

Scheduling is a decision-making process of allocating limited resources to tasks over time with the goal of optimizing a certain objective [9]. Scheduling problems can be solved by dynamic programming or integer programming models. Most scheduling problems do not have a polynomial time algorithm and are NP-hard problems. Pinedo [9] presented a complexity hierarchy of scheduling problems. The IP model for scheduling problems contains much symmetry, which makes the model hard to be solved by branch-and-bound. Lustig and Puget [3] pointed out that constraint programming is often better than integer programming in application to sequencing and scheduling.

Jain and Grossmann [10] applied the logic-based Benders cut into a planning and scheduling problem that involves cost minimization. Harjunkoski and Grossmann [11] extended the decomposition strategy for a multistage planing and scheduling problem. Maravelias and Grossmann [12] applied the decomposition to a scheduling of batch plants formulated as the State Task Network. Hooker [13] developed the logic Benders cuts for three different objectives (minimum cost, minimum makespan and minimum total tardiness) in general planning and scheduling problems. He modeled them as three different problems, solving each of them one at a time. However, the requirements from a real world application might be that several goals need to be achieved simultaneously. Multiple objective optimization might be the case. This paper discusses how to implement logic-based Benders cuts for multiple-objective optimization in planning and scheduling problems. It can be viewed as an extension to Hooker's work.

As mentioned before, a logic-based Benders cut is based on the inference of the subproblem, so there is no standard formulation for the cut. It is problem specific. Different objectives require different logical inference for cuts, so complicated objectives might result in difficulty in cut generation. This paper proposes a logic-based Benders cut approach for a planning and scheduling problem with combined multiple objectives.

3 Integrated Solution of the Problem

As stated in section 1, an assignment and scheduling problem must be solved for a set of tasks which comprise the restoration plan. The optimal solution assigns the tasks to a workgroup and then arranges the tasks into a schedule. Each task has a due date and the objective function will include a term to minimize the amount of time each task exceeds its due date by (referred to later as tardiness). The cost and time to complete a task depends on the workgroup to which it is assigned. The objective function will include terms to minimize the cost of completing all the tasks, the total tardiness, and the time to complete the last task (the makespan). Each of these three terms of the objective function will be weighted to reflect the priorities of the decision maker. The problem is fundamentally an assignment and scheduling problem. It falls into a category of problems that have proved to be difficult to solve [10,

13]. An integrated algorithm based on a MILP master problem and several CP subproblems is presented in this paper. The idea behind it is that the master problem assigns tasks to groups; for each workgroup, a subproblem schedules the tasks assigned to it. When subproblems prove the optimal solution from master problem is also feasible (for each workgroup, a schedule which achieves the same optimality as master problem can be found with satisfaction of all constraints), the algorithm will stop and the solution from the master problem is optimal to the original problem. Otherwise, relevant cutting planes are added to the master problem and the above procedure iterates. Basically, the original problem is decomposed to several smaller problems as long as an assignment is set. Because parallel groups won't interact with each other, each subproblem can be solved individually, which causes a dramatic improvement in computational performance .

3.1 Master Problem

The Master Problem determines the assignment of tasks to workgroups. The formulation requires the following notation:

i : a task,
m : a work workgroup,
I : the set of tasks,
M : the set of workgroups,
I_m : the set of tasks assigned to workgroup m,
$x_{im} := 1$ if task i is assigned to workgroup m,
t_i : starting time of task i,
s_i : tardiness of task i, always ≥ 0, since we only consider penalty.
y : makespan, i.e., completion time of the last task,
R_m : the amount of resource bundled to workgroup m,
c_{im} : the cost of workgroup m completing task i,
p_{im} : time of workgroup m requiring to complete task i,
d_i : the due time of task i,
q_i : the workgroup that task i is assigned to,
r_{im} : the amount of resource workgroup m requires to complete task i.

The Master Problem is formulated as a mixed integer program.

$$\min_{x,s,y,t} \alpha \sum_{i \in I} \sum_{m \in M} c_{im} x_{im} + \beta \sum_{i \in I} s_i + \gamma y \tag{1}$$

$$\text{subject to} \qquad t_i + \sum_{m \in M} p_{im} x_{im} - s_i \leq d_i \quad \forall i \in I \tag{2}$$

$$t_i + \sum_{m \in M} p_{im} x_{im} \leq y \quad \forall i \in I \tag{3}$$

$$\sum_{m \in M} x_{im} = 1 \quad \forall i \in I \tag{4}$$

$$\sum_{i \in I} x_{im} p_{im} \leq y \quad \forall m \in M \tag{5}$$

$$\texttt{integer cuts} \tag{6}$$

$$x_{im} = 0 \quad \text{if } r_{im} > R_m \tag{7}$$

$$x_{im} \in \{0, 1\} \; \forall i \in I \text{ and } \forall m \in M \tag{8}$$

$$s_i \geq 0 \quad \forall i \in I \tag{9}$$

$$y \geq 0 \tag{10}$$

The master problem includes all three measures: total cost, tardiness and makespan. Weights $\alpha,\ \beta,\ \gamma$ are used to trade off among those three objectives. The objective function tries to minimize the weighted sum of them. Equation (2) ensures that task i is completed before the due time, otherwise tardiness s_i is incurred. Equation (3) ensures the makespan is no smaller than the completion time of every task. Equation (4) is an assignment constraint: each task can only be assigned to one group. Equation (5) ensures that total performing time of each group should be less than or equal to makespan. Equation (6) are integer cuts from all subproblems. These integer cuts impose more and more strict restrictions on variable $x_{im},\ s_i,\ y$ as the algorithm proceeds, and eventually drive those variables to the optimal solution of the original problem. Equation (7) ensures each task is only assigned to a workgroup which has enough resources to complete it.

Note that the master problem doesn't have sequencing constraints, which means for each task no restrictions exist on starting time, so starting times in the solution of master problem might not be feasible. Since starting times imply a schedule of tasks, the master problem can't yield feasible schedules because of infeasible starting times. Tardiness s_i and makespan y are associated with schedules, infeasible schedules result in infeasible s_i and y. Therefore, the master problem yields the optimal assignment x_{im}, optimal tardiness s_i and optimal makespan y, but their feasibility can't be guaranteed. That requires the subproblems to play a role in checking feasibility of the optimal solution from the master problem.

3.2 Subproblem (CP model)

The master problem can't guarantee what it yields is feasible to the original problem, so it only solves the problem partially. The goal of the subproblem is to examine whether this partial solution can be extended to a full solution for the original problem, i.e., to feasible schedules. When the subproblem finds the schedule which can achieve the same minimum tardiness and makespan as the master problem, the original problem is solved. The cost isn't involved in the subproblem because it is only related to the assignment result and won't change. The subproblem only checks the result of the master problem and it won't change the assignment result.

The subproblem requires as input the assignment results from the master problem. For each group, the set of tasks composing the group is determined by the master problem. That is, the assignment result x_{im} from the master problem can derive the value of q_i indicating to which group task i is assigned. For each group m, a set of tasks assigned to it, $I_m := \{i \in I : q_i = m\}$, can be determined. Then a subproblem m is modeled as:

$$\min\ \beta \sum_{i \in I_m} {s'}_i + \gamma y' \tag{11}$$

$$\text{subject to}\ \ i.end \leq y^* + y' \quad \forall i \in I_m \tag{12}$$

$$i.start \leq d_i + {s'}_i - p_{iq_i} \quad \forall i \in I_m \tag{13}$$

$$i.duration = p_{iq_i} \quad \forall i \in I_m \tag{14}$$

$$i_1 \texttt{ precedes } i_2 \qquad \forall (i_1, i_2) \in \texttt{ precedence pairs }, i_1 \neq i_2 \in I_m \tag{15}$$

$$i \texttt{ require } q_i \qquad \forall i \in I_m \tag{16}$$

$${s'}_t \geq 0 \quad \forall i \in I_m \tag{17}$$

$$y' \geq 0 \tag{18}$$

The model is a pure CP model and seeks a schedule which minimizes the weighted sum of total tardiness and makespan. Variable task i is a special object in OPL [14] and has some attributes such as starting time, ending time, duration and so on. Variable ${s'}_i$ is the new tardiness. Compared with s_i from the master problem, it is the exact tardiness since the subproblem imposes effective constraints on it. Parameter y^* is the optimal makespan of the master problem. Variable y' is the slack in the makespan of the master problem, as defined in Equation (12). Equation (13) ensures every task is completed before its due time. Equation (14) specifies the processing time of task i. Equation (15) is a precedence constraint. A workgroup m is an unary resource. Equation (16) is a unary resource constraint and makes sure no two tasks requiring it are scheduled at the same time.

3.3 Algorithm

Let ${x^k}^*$, ${s^k}^*$, ${y^k}^*$ be the optimal solution of the master problem at iteration k. For each group m, a subproblem is formulated. So the number of subproblems is the number of groups and the original problem is decomposed into several small size problems. They determine our three objectives cost, tardiness and makespan, respectively. Let ${s'}^{(m,k)^*}$, ${y'}^{(m,k)^*}$ be the optimal solution of subproblem m at iteration k. Different schedules in subproblems won't change the optimal cost, because the cost is only related to the assignment. Thus, all we concerned with at iteration k is whether or not the minimum tardiness and makespan from the master problem can also be achieved in the subproblem. That is, whether the sum of the differences between the master problem and

the subproblem, i.e., difference in tardiness and difference in makespan, equals zero. As mentioned before, constraints on those two terms in the subproblem are always tighter than those in the master problem, consequently those two values in the subproblem are larger than those in the master problem. Let's introduce a new term defined as follows:

$$D^{(m,k)} = \sum_{i \in I_m^k} (s')_i^{(m,k)^*} - \sum_{i \in I_m^k} s_i^{k^*} + (y')^{(m,k)^*} \tag{19}$$

where $I_m^k = \{i \in I : {x_{im}^k}^* = 1\}$.

The *subproblem difference* $D^{(m,k)}$ is a measure of the difference in tardiness and makespan between the master problem solution and the subproblem solution. When $\sum_{m \in M} D^{(m,k)} = 0$, at iteration k the minimum tardiness and makespan from the master problem can also be achieved by optimal schedules from the subproblems, so the optimal solution for the original problem, i.e, an optimal assignment and schedules, is obtained. Otherwise, the current optimal solutions from master problem ${x_{im}^k}^*,\ s_i^{k^*},\ y^{k^*}$ needs to be cut off. Different values of those two measurements determine the cutting plane. There are four cases:

1. Subproblem m is infeasible, indicating that the assignment from the master problem is incorrect, so the cutting plane at k iteration is to cut off current assignment:

$$\sum_{i \in I_m^k} x_{im} \leq \sum_{i \in I_m^k} {x_{im}^k}^* - 1 \tag{20}$$

2. $D^{(m,k)} > 0$ indicates that tardiness $s_i^{k^*}$ and makespan y^{k^*} from the master problem might not be large enough, so cutting plane at iteration k either cuts off the current assignment or increases tardiness and makespan:

$$\sum_{i \in I_m^k} x_{im} \leq \sum_{i \in I_m^k} {x_{im}^k}^* - 1$$

or

$$\sum_{i \in I_m^k} \beta s_i + \gamma y \geq \sum_{i \in I_m^k} \beta (s')_i^{(m,k)^*} + \gamma (y^{k^*} + (y')^{(m,k)^*}) \tag{21}$$

3. $D^{(m,k)} \leq 0$ indicates that current solution from the master problem is feasible for subproblem m.
4. $\sum_{m \in M} D^{(m,k)} = 0$ holds for all subproblems, then the current solutions are optimal. The optimal assignment is from the master problem and optimal schedule is from the subproblems.

There are two fundamental constraints comprising the above cuts. Constraint (20) cuts off the current assignment by restricting the sum of assignment variables $x_{im},\ i \in I_m^k$ to be less than the current value. This cut

only involves the assignment variables currently assigned to a group, so it is tight enough to make the feasible region shrink efficiently. Constraint (21) increases the tardiness and makespan by imposing a lower bound, i.e., the optimal weighted sum of them from the current subproblem, which implies the weighted sum of tardiness and makespan should be at least as large as $\sum_{i \in I_m^k} \beta {s'}_i^{(m,k)^*} + \gamma(y^{k^*} + {y'}^{(m,k)^*})$. In other words, constraint (21) employs a nice bound to drive variable $s_i,\ i \in I_m^k$ and y towards optimality.

The logic behind the cut is when the optimal solution of the master problem is proved to be infeasible, there are two possibilities: 1) the assignment is wrong; or 2) the assignment is correct but makespan and tardiness are not large enough, which results in infeasibility. So the cut is either to cut off the assignment or to increase the makespan and tardiness. Hooker's cut [13] tried to build up the relationship between the objective value and the assignment variables. When the problem has one of the classical single objectives of cost or makespan or tardiness, it is possible to place bounds on how the objective value is changed as the values of the assignment variables change. Thus, Hooker is able to derive more specialized cuts for these objectives. However, when the objective function is a convex combination of the classical objectives, it is hard to derive such a relationship. So we use different logic to derive the cut and drive the solution to optimality. Nonetheless, as will be seen in the section on computational results, these general cuts are still powerful and enable the fast solution of realistic problems.

We use disjunctive constraints to represent this logic for case 2. The binary variable $z^{(m,k)}$ is involved in the formulation of disjunctive constraints. Whether constraint (20) and constraint (21) are effective or not is dependent on values of this binary variable. So case (2) can be rewritten as follows:

$$\sum_{i \in I_m^k} x_{im} \leq \mathsf{M} z^{(m,k)} + \sum_{i \in I_m^k} {x_{im}^k}^* - 1 \tag{22}$$

$$\sum_{i \in I_m^k} \beta s_i + \gamma y \geq z^{(m,k)} \left[\sum_{i \in I_m^k} \beta {(s')_t^{(m,k)}}^* + \gamma(y^{k^*} + {(y')^{(m,k)}}^*) \right] \tag{23}$$

M is equal to the total number of tasks, which is an upper bound which is large enough for the sum of some assignment variables, since all assignment variables are binary. When $z^{(m,k)}$ takes a value of 0, constraint (20) will take effect. Otherwise, the cut is invalidated by a redundant upper bound for x_{im}. Likewise, both variable s_i and y have a lower bound 0. When $z^{(m,k)}$ takes a value of 1, constraint (21) will take effect. When it takes a value of 0, the cut will turn out to be a redundant lower bound for s_i and y.

Figure 1 depicts the whole solving procedure. The original problem is divided into two parts, the assignment problem and the scheduling problem. The former is the master problem and includes all assignment constraints. As the algorithm proceeds, more and more integer cuts are added to it. The latter is the subproblem and includes all sequencing constraints. The algorithm

begins with solving a master problem, solutions of the master problem specify each subproblem. If a subproblem is infeasible, the cut which cuts off the current assignment will be added to the master problem, and the algorithm goes back to the beginning. If the subproblem is feasible, its optimal value will be compared with the corresponding part of master problem value. The algorithm ends up with equality of those two values. If they are not equivalent, disjunctive cuts will be added to the master problem and the algorithm goes back to the beginning. Briefly, subproblems at each iteration check if the master problem solution can be extended to feasible schedules. If so, the optimal solution for the original problem is found. Otherwise, cutting planes are added to the master problem.

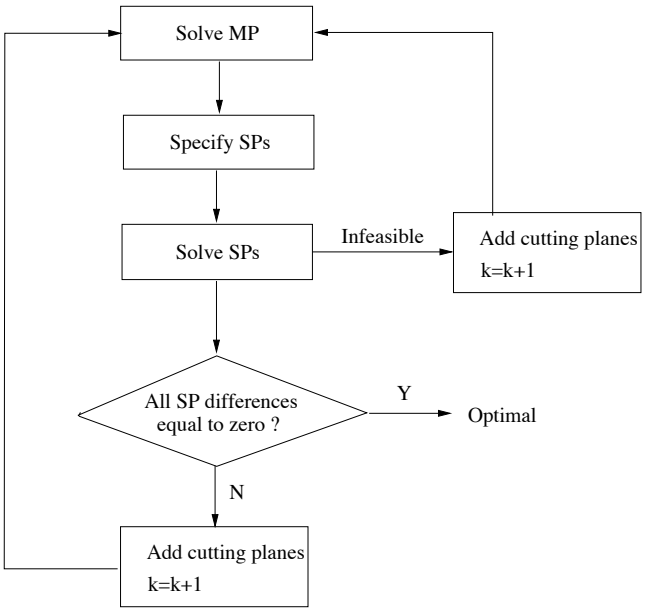

Fig. 1. Decomposition Algorithm

4 Computational Results

All CP subproblems were solved by ILOG Solver and ILOG Scheduler [15]. All MIP master problems were solved by ILOG CPLEX 8.0 [16]. The decomposition algorithm was implemented using the script language in ILOG OPL Studio 3.7 [14]. There are 3 workgroups and 30 tasks in our problem.

4.1 Objective Function Scaling

As defined in equation (1), the goal of our model is to find the optimal solution for the combined objectives (the cost, the tardiness and the makespan). They are not comparable because of the different units they use, so it is hard to reflect the priorities of the decision maker by weights. Our approach to address this issue is to convert different units of measures into the same unit. The conversion can be viewed as a process of finding the relationship of different units, i.e., how much does it cost when there is a one day delay? How much does it cost for every working day? We try to measure makespan and tardiness by money. Therefore, the objective can be rewritten as follows:

$$\alpha \sum_{i \in I} \sum_{m \in M} c_{im} x_{im} + \beta \eta_1 \sum_{i \in I} s_i + \gamma \eta_2 y \tag{24}$$

where η_1 is the ratio of the cost to the tardiness, η_2 is the ratio of the cost to the makespan.

Generally, ratios η_1 and η_2 need to be set by model users. Sometimes, their values can be derived mathematically. Take η_2 as a example, set α to be 100, β to be 0 and γ to be 0, solve the problem for the minimum cost ($cost^*$), and then set α to be 0, β to be 0 and γ to be 100, solve the problem for the minimum makespan ($makespan^*$). Then η_2 can be set as follows:

$$\eta_2 = \frac{cost^*}{makespan^*} \tag{25}$$

The argument for this method is that given the same amount of resources and same constraints two goals can be achieved individually, so the ratio of their optimal values can be viewed as the ratio of the cost to the makespan for those resources and constraints. This method only applies to the problem with nonzero optimal value for each objective, otherwise, the an infinitely large ratio could be derived.

For the problem discussed in this paper, we get three extreme points by considering only one objective at a time. They are (442,240,60), (503,0,60) and (502,240,20), corresponding to three sets of (α, β, γ) equal to (100,0,0), (0,100,0) and (0,0,100), respectively. So, in order to complete all restoration tasks, we need to pay at least 442 dollars, which is the lower bound of the cost. If we don't care about how much completing the task will cost or how long completing all tasks will take, then we can complete all tasks by its due date. If we don't care about the cost or if every task is done by the due date, then we can finish all the tasks in 20 days.

$$\begin{aligned} \text{cost}^* &= 442 \\ \text{tardiness}^* &= 0 \\ \text{makespan}^* &= 20 \end{aligned}$$

The value of η_2 is obtained by dividing $cost^*$ by $makespan^*$, giving 22. While the above method doesn't apply to η_1 since $tardiness^*$ is 0, it is set to be 1 by user's experience.

4.2 Computational Performance

Table 1 shows computational performance for different values of weight. *# cuts* gives the number of cuts generated to solve the problem. *Iterations* lists the number of times the Master Problem was solved. *Solving time* is the CPU time for solving the model. It was calculated by summing up solving times of the MIP model and 3 CP models. Different combinations of the three weights α, β, γ lead to different complexities of problems. As the proportion of α decreases, deviation of the master problem solution from the optimal solution to the original problem becomes larger. Thus more iterations are involved in the algorithm.

Table 1. Computational performance of algorithm for different combinations of α, β, γ.

No.	α	β	γ	# Cuts	Iterations	Solving time (seconds)
1	0	75	25	28	22	1.834
2	5	70	25	28	21	1.812
3	10	65	25	27	20	1.545
4	15	60	25	27	20	1.672
5	20	55	25	21	17	1.187
6	25	50	25	18	15	0.766
7	30	45	25	14	12	0.547
8	35	40	25	14	12	0.673
9	40	35	25	10	9	0.500
10	45	30	25	9	8	0.391
11	50	25	25	7	6	0.172
12	55	20	25	5	5	0.156
13	60	15	25	2	3	0.079
14	65	10	25	2	3	0.078
15	70	5	25	2	3	0.078
16	75	0	25	0	1	<0.001

4.3 Optimal Solutions

Tables 2–10 display optimal solutions for different priorities on three objectives: cost, tardiness and makespan. Almost all of the problems could be solved in less than four seconds. The exceptions are when the makespan has low priority, in particular γ is equal to 5 or smaller. We found that the problem could

not be solved in two hours for many of these priority combinations with $\gamma = 0$ so these results have been omitted. The sum of the three weights is 100. Each weight is held constant at 3 different levels, i.e., 25, 50 and 75, and the other two weights varied by multiples of five. It can be seen that most of the time optimal solutions don't change as weights change. This uneven distribution of optimal solutions was noted by Das and Dennis [17]. They pointed out that an evenly distributed set of weights fails to produce an evenly distributed set of points from all parts of the Pareto set. The result set presents frontier values for different weights.

Table 2. γ held constant at 25.

No.	α	β	γ	Cost (dollars)	Tardiness (days)	Makespan (days)	Solving time (seconds)
1	0	75	25	497	3	20	1.834
2	5	70	25	495	3	20	1.812
3	10	65	25	495	3	20	1.545
4	15	60	25	495	3	20	1.672
5	20	55	25	495	3	20	1.187
6	25	50	25	495	3	20	0.766
7	30	45	25	495	3	20	0.547
8	35	40	25	495	3	20	0.673
9	40	35	25	494	4	20	0.500
10	45	30	25	494	4	20	0.391
11	50	25	25	483	3	21	0.172
12	55	20	25	483	3	21	0.156
13	60	15	25	483	3	21	0.079
14	65	10	25	483	3	21	0.078
15	70	5	25	483	3	21	0.078
16	75	0	25	483	240	21	<0.001

Let us take Table 2 as an example. In this case, the weight of makespan, γ, is held constant at 25 and five optimal solution sets are obtained by varying weights of cost and tardiness, α and β. It can be seen that when makespan is 25% of the total objective, the three efficient (non-dominated) solutions are $(495, 3, 20)$, $(494, 4, 20)$, and $(483, 3, 21)$. Those three solutions compose the frontier value set of optimal solutions at $\gamma = 25$ level. In this way, frontiers for different levels of the weights can be presented. The set of such frontiers provides the decision maker a nice picture of the correspondence between weights and optimal solutions.

Tables 3 and 4 show the efficient solutions are (495,3,20) and (494,4,20) when γ is held constant at 50 or 75. Table 5 shows the efficient solutions are (495,3,20), (494,4,20),(488,0,21) and (480,0,22) when α is held constant at 25. Table 6 shows the efficient solutions are (494,4,20), (483,3,21), (472,3,23) and (470,1,24) when α is held constant at 50. Table 7 shows the efficient

Table 3. γ held constant at 50.

No.	α	β	γ	Cost (dollars)	Tardiness (days)	Makespan (days)	Solving time (seconds)
1	0	50	50	497	3	20	1.264
2	5	45	50	495	3	20	1.391
3	10	40	50	495	3	20	1.359
4	15	35	50	495	3	20	0.906
5	20	30	50	495	3	20	0.485
6	25	25	50	494	4	20	0.328
7	30	20	50	494	4	20	0.110
8	35	15	50	494	4	20	0.156
9	40	10	50	494	4	20	0.094
10	45	5	50	494	4	20	0.063
11	50	0	50	494	240	20	0.016

Table 4. γ held constant at 75.

No.	α	β	γ	Cost (dollars)	Tardiness (days)	Makespan (days)	Solving time (seconds)
1	0	25	75	495	3	20	1.001
2	5	20	75	495	3	20	1.609
3	10	15	75	495	3	20	0.469
4	15	10	75	494	4	20	0.172
5	20	5	75	494	4	20	0.094
6	25	0	75	494	240	20	0.031

Table 5. α held constant at 25.

No.	α	β	γ	Cost (dollars)	Tardiness (days)	Makespan (days)	Solving time (seconds)
1	25	0	75	494	240	20	0.016
2	25	5	70	494	4	20	0.047
3	25	10	65	494	4	20	0.095
4	25	15	60	494	4	20	0.203
5	25	20	55	494	4	20	0.235
6	25	25	50	494	4	20	0.280
7	25	30	45	495	3	20	0.485
8	25	35	40	495	3	20	0.469
9	25	40	35	495	3	20	0.625
10	25	45	30	495	3	20	0.718
11	25	50	25	495	3	20	0.828
12	25	55	20	495	3	20	1.110
13	25	60	15	488	0	21	3.751
14	25	65	10	488	0	21	3.769
15	25	70	5	480	0	22	14.203

Table 6. α held constant at 50.

No.	α	β	γ	Cost (dollars)	Tardiness (days)	Makespan (days)	Solving time (seconds)
1	50	0	50	494	240	20	0.016
2	50	5	45	494	4	20	0.046
3	50	10	40	494	4	20	0.031
4	50	15	35	494	4	20	0.109
5	50	20	30	494	4	20	0.141
6	50	25	25	483	3	21	0.204
7	50	30	20	483	3	21	0.109
8	50	35	15	483	3	21	0.780
9	50	40	10	472	3	23	3.655
10	50	45	5	470	1	24	2923

Table 7. α held constant at 75.

No.	α	β	γ	Cost (dollars)	Tardiness (days)	Makespan (days)	Solving time (seconds)
1	75	0	25	483	240	21	0.031
2	75	5	20	483	3	21	0.140
3	75	10	15	471	7	23	0.062
4	75	15	10	461	7	26	3.765
5	75	20	5	452	14	30	500

Table 8. β held constant at 25.

No.	α	β	γ	Cost (dollars)	Tardiness (days)	Makespan (days)	Solving time (seconds)
1	0	25	75	495	3	20	1.077
2	5	25	70	495	3	20	1.344
3	10	25	65	495	3	20	0.657
4	15	25	60	495	3	20	0.734
5	20	25	55	495	3	20	0.658
6	25	25	50	494	4	20	0.265
7	30	25	45	494	4	20	0.282
8	35	25	40	494	4	20	0.250
9	40	25	35	494	4	20	0.172
10	45	25	30	494	4	20	0.187
11	50	25	25	483	3	21	0.266
12	55	25	20	483	3	21	0.110
13	60	25	15	472	3	23	1.703
14	65	25	10	472	3	23	4.453
15	70	25	5	452	14	30	1998

Table 9. β held constant at 50.

No.	α	β	γ	Cost (dollars)	Tardiness (days)	Makespan (days)	Solving time (seconds)
1	0	50	50	497	3	20	1.297
2	5	50	45	495	3	20	1.375
3	10	50	40	495	3	20	1.313
4	15	50	35	495	3	20	1.718
5	20	50	30	495	3	20	0.875
6	25	50	25	494	4	20	0.797
7	30	50	20	494	4	20	1.342
8	35	50	15	483	3	21	1.156
9	40	50	10	480	0	22	6.705
10	45	50	5	470	1	24	724

Table 10. β held constant at 75.

No.	α	β	γ	Cost (dollars)	Tardiness (days)	Makespan (days)	Solving time (seconds)
1	0	75	25	496	3	20	1.844
2	5	75	20	495	3	20	2.250
3	10	75	15	495	3	20	2.220
4	15	75	10	488	0	21	2.751
5	20	75	5	480	0	22	3.111

solutions are (483,3,21), (471,7,23), (461,7,26) and (452,14,30) when α is held constant at 75. Table 8 shows the efficient solutions are (495,3,20), (494,4,20), (483,3,21), (472,3,23) and (452,14,30) when β is held constant at 25. Table 9 shows the efficient solutions are (495,3,20), (494,4,20), (483,3,21), (480,0,22)and (470,1,24) when β is held constant at 50. Table 10 shows the efficient solutions are (495,3,20), (488,0,21), (480,0,22)and (470,1,24) when β is held constant at 75.

Altogether, eleven efficient solutions were found, and ten of these are displayed in Figure 2. The additional solution is the efficient solution (442,240,60) found when minimizing the single objective of cost; this is not plotted because it would change the scaling of the picture too dramatically. The solutions found by minimizing the single objectives makespan and tardiness were not efficient. As can be seen from the tables, these solutions are found very quickly, making it practical to determine them all. It appears that either an integrated MIP approach or a Benders decomposition approach where the subproblems were solved using integer programming techniques would require far more time. Because of the speed of the algorithm, all these efficient solutions could be presented to a decision maker, who could then choose an appropriate

schedule of tasks to workgroups in order to restore the interdependent layered network.

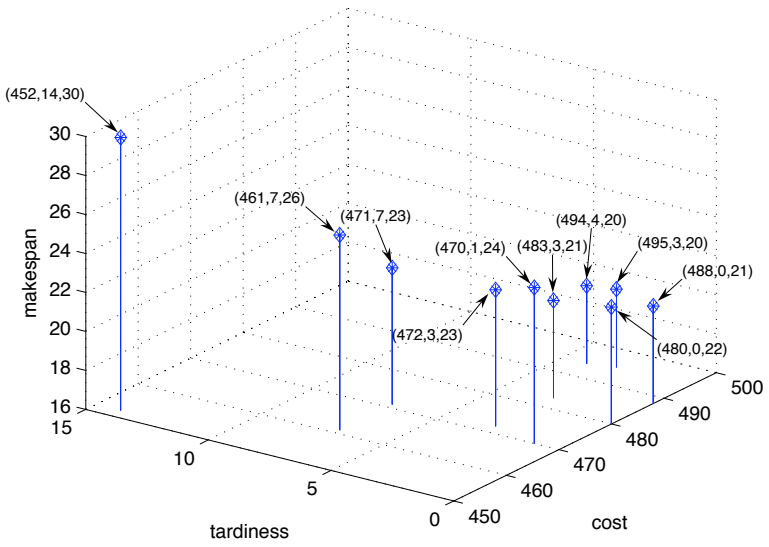

Fig. 2. Efficient points for the problem

5 Conclusions and Future Work

This paper presents a general framework of a logic-based Benders cut for objective functions that combine the cost, the tardiness and the makespan. Disjunctive cuts are generated based on logical inference. Our computational results show that the algorithmic framework allows rapid solution of these problems, enabling the determination of representative points on the efficient frontier set of optimal solutions for a multiple objective optimization problem.

Future work will address the following issues:

1. **Take shared resources into account**. As described earlier, the resources considered in the problems are bundled into workgroups. i.e. unary resources. When shared resources are considered, each subproblem can't be solved individually. The shared resources must be considered across all the subproblems. This will imposes a challenge on the decomposition algorithm and cutting planes.
2. **Integrate determination of restoration plan with assignment and scheduling**. The example given separates the determination of the

restoration plan from its cost and schedule. Solving the assignment and scheduling problem is the second step in restoration. Separation of the procedure into two steps could result in a case where the optimal restoration plan found in the first step is hard to implement in the second step for some reason, for example, limited budget, limited resources, and so on. Integration of the two steps into a single process might yield more efficient restoration plans. The speed of solution of the scheduling problem should make this integration possible.

6 Acknowledgments

This research is supported by NSF grant CMS 0301661, Decision Technologies for Managing Critical Infrastructure Interdependencies

References

1. E.E. Lee, J.E. Mitchell, W.A. Wallace (2006) Restoration of Services in Interdependent Infrastructure Systems: A Network Flows Approach. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, to appear.
2. E.E. Lee (2006) Assessing Vulnerability and Managing Disruptions to Interdependent Infrastructure Systems: A Network Flows Approach. Ph.D. Thesis, Rensselaer Polytechnic Institute, New York
3. I.J. Lustig, J.F. Puget (2001) Program Does Not Equal Program: Constraint Programming and Its Relationship to Mathematical Programming. Interfaces 31:29-53
4. P.V. Hentenryck, L. Perron, J.F. Puget (2000) Search and Strategies in OPL. ACM Transactions on Computational Logic 1:282–315
5. H.J. Kim, J.N. Hooker (2002) Solving fixed-charge network flow problems with a hybrid optimization and constraint programming approach. Annals of Operations Research 115:95–124
6. J.N. Hooker (2002) Logic, optimization and constraint programming. INFORMS Journal on Computing 14:295–321
7. J.N. Hooker (2005) A Search-infer-and-relax Framework for Integration Solution Methods. Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems(CPAIOR), LNCS 3524:243-257
8. J.N. Hooker (2000) Logic-based Methods for Optimization: Combining Optimization and Constraint Satisfaction. John Wiley, New York
9. M. Pinedo (2002) Scheduling: Theory, Algorithms and Systems. Prentice Hall, Upper Saddle River, New Jersey
10. V. Jain, I.E. Grossmann (2001) Algorithms for Hybrid MILP/CP Models for A Class of Optimization Problems. INFORMS Journal on Computing 13:258–27
11. I. Harjunkoski, I. E. Grossmann (2002) Decomposition Techniques for Multistage Scheduling Problems Using Mixed-integer and Constraint Programming Methods. Computers and Chemical Engineering 26:1533–1552

12. C.T. Maravelias, I.E. Grossmann (2004) A Hybrid MILP/CP Decomposition Approach for the Continuous Time Scheduling of Multipurpose Batch Plants. Computers and Chemical Engineering 28:1921–1949
13. J.N. Hooker (2004) Planning and Scheduling by Logic-based Benders Decomposition. Operations Research, to appear
14. ILOG Inc (2002) ILOG OPL Studio 3.7.1 Language Manual. ILOG Inc. Mountain View, California
15. ILOG Inc (2002) ILOG OPL Studio 3.7.1 User's Manual. ILOG Inc. Mountain View, California
16. ILOG Inc (2002) ILOG CPLEX 8.0 User's Manual. ILOG Inc. Mountain View, California
17. I. Das, J. Dennis (1997) A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems. Structural Optimization 14: 63-69