6-1-2016

Analysis of a Parallel Machine Scheduling Problem with Sequence Dependent Setup Times and Job Availability Intervals

Ridvan Gedik
University of New Haven, rgedik@newhaven.edu

Chase Rainwater
University of Arkansas, Fayetteville

Heather Nachtmann
University of Arkansas, Fayetteville

Edward A. Pohl
University of Arkansas, Fayetteville

Publisher Citation

Gedik, R., Rainwater, C., Nachtmann, H., & Pohl, E. A. (2016). Analysis of a parallel machine scheduling problem with sequence dependent setup times and job availability intervals. European Journal of Operational Research, 251(2), 640-650.

Comments

This is the authors' accepted version of the article published in *European Journal of Operational Research*. The version of record can be found at http://dx.doi.org/10.1016/j.ejor.2015.11.020

Analysis of a Parallel Machine Scheduling Problem with Sequence Dependent Setup Times and Job Availability Intervals

Ridvan Gedik[a,*], Chase Rainwater[b], Heather Nachtmann[b], Ed A. Pohl[b]

[a] *Department of Mechanical and Industrial Engineering*
University of New Haven
300 Boston Post Rd
West Haven, CT 06516, USA
[b] *Department of Industrial Engineering*
University of Arkansas
4207 Bell Engineering Center
1 University of Arkansas
Fayetteville, AR 72701, USA

Abstract

In this study, we propose constraint programming (CP) model and logic-based Benders algorithms in order to make the best decisions for scheduling non-identical jobs with availability intervals and sequence dependent setup times on unrelated parallel machines in a fixed planning horizon. In this problem, each job has a profit, cost, has to be assigned to at most one machine in such a way that total profit is maximized. In addition, the total cost has to be less than or equal to a budget level. Computational tests are performed on a real-life case study prepared in collaboration with the U.S. Army Corps of Engineers (USACE). Our initial investigations show that the pure CP model is very efficient in obtaining good quality feasible solutions but, fails to report the optimal solution for the majority of the problem instances. On the other hand, the two logic-based Benders decomposition algorithms are able to obtain near optimal solutions for 86 instances out of 90 examinees. For the remaining instances, they provide a feasible solution. Further investigations show the high quality of the solutions obtained by the pure CP model.

Keywords: Constraint programming, mixed integer programming, logic-based Benders decomposition, scheduling with job availability intervals, sequence dependent setup times

1. Introduction

This study aims to provide novel modeling and solution techniques for the problem of managing a fleet of resources that is composed of a limited number of unrelated machines subject to operational and tactical level restrictions. For a given time horizon, we seek to assign non-identical jobs to machines and provide the best sequence of jobs on each machine. During the search for the best assignment and job sequence, one must also account for the time spent in

*Corresponding author, Tel:+1 (203) 932-1167
Email addresses: rgedik@newhaven.edu (Ridvan Gedik), cer@uark.edu (Chase Rainwater), hln@uark.edu (Heather Nachtmann), epohl@uark.edu (Ed A. Pohl)

between jobs due to setup or travel. Moreover, the procession time of each job depends on the type of the machine assigned to it. Each job has a cost, profit, availability interval(s) and can be assigned to *at most* one machine. The total cost of operations across all jobs has to be less than or equal to a known budget level. Subject to all these requirements, a typical decision-maker's objective is to maximize the total profit within a given time horizon. In order to clarify some other aspects of the problem description, the following assumptions are made for all models given in this study:

- Release time and deadline of each job are start and end of planning horizon, respectively. A job cannot be completed after the end of planning horizon.
- All jobs are available for processing at the beginning of time horizon.
- A machine can work on at most one job at a time.
- Job preemption is not allowed.
- Job processing times are deterministic, but vary based on the machine type.
- Sequence dependent travel (setup) time between jobs is deterministic and assumed to be same for each machine.
- There is no travel (setup) time before the first job and after the last job in a machine's schedule.
- A job might have more than one availability interval. A *restricted period* is defined as a complementary concept for availability interval to represent the times when job processing is prohibited.
- Due to limited resources and time, jobs are selective. In other words, a job can be assigned to at most one machine.

The remainder of the paper is organized as follows. Section 2 discusses the integer programming (IP), constraint programming (CP) and decomposition algorithms developed to solve the scheduling problem defined above. A real world application of this problem with corresponding instances is introduced in Section 3. Finally, Section 4 compares the performance of the proposed solution techniques.

2. Problem Modeling: Mathematical versus Constraint Programming

The scheduling problem defined in Section 1 shares similar properties with the operational fixed interval or job scheduling problem (OFISP or OFJSP). OFJSP aims to schedule number of jobs with a *fixed or known* start and end time on parallel resources in order to maximize the total weights (profits) by processing a subset of jobs (Kroon et al. (1995)). OFJSP is a generalized version of the Fixed Job Scheduling Problem (FSP) in which the objective is to find a feasible non-preemptive schedule of jobs with fixed start and end times on identical parallel machines. Both OFJSP and the problem of interest in this study assume that (i) no preemption is allowed, (ii) jobs are selective and (iii) at most one job can be processed by a machine at a given time. Aside from these three common fundamental properties, the proposed problem in this study differs from OFJSP. OFJSP assumes that the processing times are fixed and can be calculated by subtracting *fixed start time* from *fixed end time*. Moreover, the processing time of a job does not vary based on the machine assignment in OFJSP. On the other hand, the investigated problem in this study seeks to find the best start and

end times to process a job with varying processing times due to unrelated parallel resources. Additionally, it accounts for job availability interval restrictions, sequence dependent setup times between jobs and a budget constraint.

Arkin and Silverberg (1987) developed a polynomial algorithm for the fixed job scheduling problem (FSP) and showed that the FSP with unrelated machines is NP-complete. Later on, Kroon et al. (1995) designed an exact algorithm for the OFJSP with the single machine based on the polynomial time algorithm in Arkin and Silverberg (1987) and developed two dual-cost heuristics that are able to find good solutions with 0-7% optimality gap for the OFJSP problem instances with unrelated machines and non-identical jobs. Türsel Eliiyi and Azizoğlu (2009) studied FSP with machine dependent job values (weights) and developed a branch and bound algorithm that is capable of obtaining optimal solutions for large problem instances. Türsel Eliiyi and Azizoğlu (2010, 2011) further proposed efficient solution techniques for the OFJSP with working and spread time constraints where working time constraints limit the total processing load and spread time constraints limit the time between the start of the first job and the finish of the last job on each machine.

Rojanasoonthon and Bard (2005) addressed a parallel machine scheduling problem with time windows and priority levels on jobs under the objective of maximizing the number of jobs scheduled. They reported that their integer linear programming (ILP) model is unable to report even a feasible solution and therefore, they provided a greedy randomized adaptive search procedure that produced good quality solutions for the data instances with 400 jobs and 6 machines. Pearn et al. (2002) studied a parallel machine scheduling problem in a wafer probing factory with non-identical jobs, identical machines and sequence dependent setup times between jobs. They proposed an ILP model and its transformation to a well studied vehicle routing problem with time windows. Three different heuristics were used to find the near-optimal solutions for real-world test instances. Cakici and Mason (2007) also considered parallel scheduling problem in semiconductor manufacturing with auxiliary resource constraints and proposed a heuristic that produced solutions with 0.78% optimality gap.

In recent years, constraint programming has been widely applied to a variety of scheduling problems as an alternative or in conjunction with mathematical programming (MP). One of the fundamental differences between CP and MP is the way that they express constraints and define decision variables to solve an optimization or a feasibility problem. On one hand, CP allows declarative, flexible and compact formulations that make adding new constraints to the search more straightforward (Focacci et al. (2002); Hooker (2007a)). Especially, without the need for a reformulation, being able to represent complex relationships (i.e. if-then, nonlinearity, logical conditions, etc.) in terms of *global (logical) constraints* is the competitive advantage of CP over MP in finding good quality feasible solutions (Jain and Grossmann (2001)). Thus, CP has been very effective in tackling highly constrained discrete optimization and feasibility problems such as scheduling, planning and resource allocation. On the other hand, MP is more likely to perform better than CP when the tackled problem has a pure (well studied) geometrical structure (Focacci et al. (2002); Lombardi and Milano (2012)). However, as the side constraints are added to the problem, the structure becomes less pure and solving MP models becomes more challenging. In this case, the application of CP becomes more appropriate. A recent increase in

the number of studies which employ CP as a (part of) solution technique for many different combinatorial optimization problems demonstrates a growing interest for this modeling technique.

Jain and Grossmann (2001) compared pure CP and MP models with a hybrid CP/MP model on a scheduling problem that involves unrelated parallel machines, non-identical orders with different release times and deadlines. They demonstrated that the hybrid model is able to solve larger instance as opposed to pure CP and ILP models. Besides, the decomposition (logic-based Benders) algorithm outperforms all other alternatives in terms of solution time and objective function value. Later, Sadykov and Wolsey (2006) studied the same problem and generated efficient decomposition algorithms based on column generation (branch and price) and branch and cut approaches. Harjunkoski and Grossmann (2002) also studied the parallel scheduling problem motivated by Jain and Grossmann (2001) with sequence independent setup times and developed another decomposition method in which both master and subproblems are formulated as ILP. More details on CP/MP based decomposition algorithms and hybrid modeling approach are given in Section 2.3.

For some other versions of parallel machine scheduling problems, a double or combined modeling approach is reported to be an efficient solution method. Edis and Ozkarahan (2011) developed a combined CP/MP model formulation which tackles "resource-constrained identical parallel machine scheduling problem with machine eligibility restrictions" and showed that the combined CP/MP model is able to find the optimal solution in most of the test instances with substantial improvements in solution time as opposed to pure ILP and CP models. Similar success of the combined CP/MP approach is reported by Edis and Oguz (2012) who tackled parallel the machine scheduling problem with flexible resources. We also see successful implementations of the combined CP/MP approach over some other operations research problems such as time-tabling/rostering (Topaloglu and Ozkarahan (2011); He and Qu (2012)), sport scheduling (Trick and Yildiz (2007, 2011)) and project scheduling with time windows (Cesta et al. (2002)). For further information about double or hybrid modeling and other decomposition methods, we refer the reader to Hooker (2006b) and Van Hoeve and Katriel (2006).

2.1. Integer Programming (IP) Formulation

The necessary notation for the scheduling problem is given in Table 1.

Van den Akker et al. (2000) pointed out that LP relaxations of the time-indexed IP formulations provide good bounds for the overall scheduling problem although it generates more decision variables and constraints than a classic sequence-based formulation. We introduce the time-indexed mixed integer programming model (DS) as follows.

Table 1: Notation

Sets	
D	set of machines
T	set of consecutive time periods comprising the planning horizon
J	set of jobs
W_j	set of restricted periods applicable to job $j \in J$
Parameters	
b_w	the beginning of restricted period $w \in W_j$; $j \in J$
e_w	the end of restricted period $w \in W_j$; $j \in J$
r_d	the operation rate of machine $d \in D$
q_j	the profit associated with job $j \in J$
p_{jd}	the time it takes for machine $d \in D$ to process job $j \in J$
$t_{jj'}$	the travel or setup time between job $j \in J$ and job $j' \in J$ ($j \neq j'$)
c_j	the cost for completing job $j \in J$
B	the available budget for the planning horizon
Decision variables	
y_{dj}	1 if machine d is used to complete job j
z_{djt}	1 if machine d begins working on job j in period t

$$\text{maximize} \sum_{j \in J} \sum_{d \in D} q_j y_{dj}$$

subject to (DS)

$$\sum_{d \in D} y_{dj} \leq 1 \qquad j \in J \tag{1}$$

$$\sum_{j \in J} \sum_{d \in D} c_j y_{dj} \leq B \tag{2}$$

$$\sum_{t \in T} z_{djt} = y_{dj} \qquad j \in J;\ d \in D \tag{3}$$

$$\sum_{t'=t}^{\min\{|T|, t+p_{jd}+t_{jj'}\}} z_{dj't'} \leq 1 - z_{djt} \qquad j \in J;\ j' \in J;\ j \neq j';\ d \in D;\ t \in T \tag{4}$$

$$\sum_{d \in D} \sum_{t=\max\{1, b_w - p_{jd}\}}^{e_w} z_{djt} = 0 \qquad w \in W_j;\ j \in J \tag{5}$$

$$(t + p_{jd})\, z_{djt} \leq |T| \qquad j \in J;\ d \in D;\ t \in T \tag{6}$$

$$y_{dj} \geq 0 \qquad d \in D;\ j \in J \tag{7}$$

$$z_{djt} \in \{0, 1\} \qquad d \in D;\ j \in J;\ t \in T \tag{8}$$

The objective of the DS model is to maximize the total profit. Constraints (1) ensure that job j is processed by

at most one machine d, whereas constraint (2) states that the total cost incurred by such assignment cannot exceed the total budget. Constraints (3) assure that job j has a start time if it is processed by machine d. Constraints (4) specify that if machine d starts processing job j in period t, then machine d cannot start working on another job, j', until $t_{jj'} + p_{jd}$ periods have passed (i.e. the time to complete job j on machine d plus the time to travel to job j' from job j). Constraints (5) prevent a job from being processed in a time period that overlaps with its restricted period(s). Constraints (6) ensure that the completion time of a job cannot exceed the end of the planning horizon. Finally, constraints (7)-(8) specify the appropriate domain of each variable in the model.

As with many integer programs for scheduling problems, providing the exact optimal schedule for each machine gets more challenging as the number of decision variables and constraints increases. Given the difficulty in solving the problem DS in an IP context and the strengths of CP in finding feasible solutions for highly constrained problems, next two sections discuss the equivalent CP model formulation of the DS and propose two decomposition algorithms that possess the competitive advantages of both CP and IP.

2.2. Constraint Programming Approach

2.2.1. Search in Constraint Programming

The effectiveness of CP is highly correlated with the constraint and variable definition choices a modeler makes. Heipcke (1999) pointed out that CP models, in general, carry much more specific information about decision variables, constraints and the relationships between/among them. This creates a flexible environment for developing stronger, more efficient and specialized solution strategies for highly constrained complex problems. Conveying information between constraints and variables is made possible through *constraint propagation (filtering)* iterative process of global constraints. Each global constraint is associated with a propagation algorithm that is used to remove the values of variables from their domains in order to prevent constraints from being infeasible (Van Hoeve and Katriel (2006); Hooker (2006b)). The propagation algorithm of a constraint is called each time a change occurs on a variable. The constraints are related to each other through shared variables. Whenever a change occurs on the domain of a shared variable due to the propagation algorithm of a constraint, the filtering algorithms of other constraints are also triggered to evaluate possible other reductions in the domains of all variables (Lombardi and Milano (2012); Harjunkoski and Grossmann (2002); Van Hoeve and Katriel (2006)). Once all possible reductions on domains are made and yet a feasible solution has not been found, branching on a variable takes place. At this point, one can see that addition of new constraints does not impact the search since the propagation algorithms of the previous constraints will remain unchanged due to *incremental search* (Focacci et al. (2002)). In other words, the newly added constraints due to branching interact with the previous ones through the shared variables and the propagation algorithms might have already reduced their domains. Thus, the order in which the constraints are propagated is irrelevant to finding a feasible solution within the iterative constraint propagation process.

There are two main properties of a CP problem formulation that might be useful in speeding up the search pro-

cess. First one is the size of the variable domains. The tighter the variable domains are defined, the less number of candidate values is going to be evaluated for feasibility by the propagation algorithms. Secondly, the constraints should be specified by the most appropriate global constraints available for the application. Each global constraint has a special filtering algorithm that might be triggered several times during a search. Since these algorithms are designed specifically for a global constraint, a failure in global constraint selection may lead to less effective propagation (Van Hoeve and Katriel (2006); Focacci et al. (2002)). This is because, employing global constraints handles *non-primitive* constraints in CP (Focacci et al. (2002); Hooker (2002); Heipcke (1999)). In both ILP and CP, *primitive* constraints are those that are easily handled, while *non-primitive* ones are those for which no complete method exists for satisfiability in polynomial time (Focacci et al. (2002)). In CP, primitive constraints are $\leq, \geq, \neq, =$, and $integrality$, while all other constraints are non-primitive. Since constraint propagation algorithms of non-primitive constraints enforce tightening on variable domains, complex constraints should always be formulated in terms of a global constraint, the propagation algorithm of which is proved to be efficient (Lombardi and Milano (2012), Hooker (2006b), Focacci et al. (2002)). Finally, Hooker (2002) pointed out that CP is more effective than MP if the constraints do not contain too many variables.

2.2.2. Constraint Programming Model

In addition to the notation in Table 1, the following parameters and decision variables are used in developing the CP formulation.

Parameters

- $I(j)$ is the *step function* of job $j \in J$. That is $I(j) = 0\%$, if the job j is not allowed to be processed at time t such that $b_w \leq t \leq e_w$, $I(j) = 100\%$ otherwise.
- $TD(t_{jj'})$ is the *transition distance function* between job $j \in J$ and $j' \in J$. It is used to inform other global constraints that the travel time between job pairs j and j' should be at least $t_{jj'}$.

An *interval variable* (IBM (2014)) is a powerful way of representing generic decision variables of a scheduling problem. As depicted in Figure 1, it addresses the time interval of a job that is being processed by explicitly assigning start and end times. One of its important features is that these variables can be *optional* which enables modeling different assignment alternatives in combinatorial problems. For instance, if an interval variable is optional and *absent*, it is not considered in the solution schedule and its domain is left empty. Otherwise, if an optional interval variable is *present*, it implies that it is considered in the solution and its domain should be filtered to a single value represented by a start and end time. The status or Boolean value of an interval variable can be retrieved by using the *presenceOf(Interval Variable)* constraint. In light of the basic definition of an interval variable, we define the following interval decision variables and sets.

Decision variables and sets

- Y_{jd}, optional interval variable when job $j \in J$ is assigned to machine $d \in D$ with job duration of p_{jd};
- Z_j, optional interval variable associated with job $j \in J$;
- $Q_j = \{Y_{j1}, Y_{j2}, \ldots, Y_{j|D|}\}$, set of interval variables representing possible machine $d \in D$ that can be assigned to job $j \in J$;
- $V_d = \{Y_{1d}, Y_{2d}, \ldots, Y_{|J|d}\}$, set of interval variables representing possible jobs $j \in J$ that can be assigned to machine $d \in D$ (*interval sequence variable* for d).

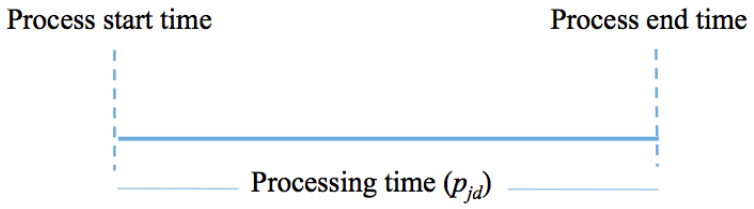

Figure 1: Interval variable representation

The constraint programming formulation of the parallel machine scheduling problem with maximizing total profit (CP-DS) is given below.

$$\text{maximize} \quad \sum_{j \in J} q_j presenceOf(Z_j)$$

subject to (CP-DS)

$$Alternative(Z_j, Q_j) \qquad j \in J \tag{9}$$

$$Cumulative(Z_j, c_j, B) \tag{10}$$

$$Cumulative(Z_j, 1, |D|) \tag{11}$$

$$Z_j.StartMin = 1 \qquad j \in J \tag{12}$$

$$Z_j.EndMax = |T| \qquad j \in J \tag{13}$$

$$ForbidExtent(Z_j, I(j)) \qquad j \in J \tag{14}$$

$$NoOverlap(V_d, TD(t_{jj'})) \qquad d \in D \tag{15}$$

The objective function of (CP-DS) seeks to maximize the total profit in a planning horizon. Constraints (9) ensure that each job can only be assigned to at most one machine. $Alternative$ constraints enforce that if Z_j is present in the solution (i.e. job j is to be processed), then only one of the elements of $Q_j = \{Y_{j1}, Y_{j2}, \ldots, Y_{j|D|}\}$ will be present in the solution in order to assign job j to a machine in D. Note that this assignment will also attribute a final processing time (p_{jd}) to Z_j. On the other hand, if Z_j is not present in the solution (i.e. job j is not to be processed), Constraints (9) guarantee that none of the elements of Q_j will be present in the solution. Constraint (10) assures that the total

cost of operations cannot exceed the budget. *Cumulative* constraint is used to model the resource usage over time and computed with the help of its elementary sub-functions such as $Step$, $Pulse$, $StepAtStart$ and $StepAtEnd$ (IBM (2014)). $StepAtStart(Z_j)$ is used to increase the total money spent on operations at the start of interval variable Z_j by c_j amount. *Cumulative* in constraint (10) is utilized for restricting total spending not to exceed the budget at any time. Similarly, the *Cumulative* constraint and $Pulse(Z_j)$ function are used to make sure that total number of occupied machines at any time can not exceed the fleet size ($|D|$) as in constraint (11) where $Pulse(Z_j)$ increases and decreases the cumulative usage of fleet by one at the start and end of interval variable Z_j, respectively.

Constraints (12) and (13) set the minimum start time and maximum end time of each job to the first and last day of the planning horizon, respectively. *ForbidExtent* constraint (14) states that if interval variable Z_j is present in the solution, it cannot overlap with the time intervals where its step function is 0%.

NoOverlap constraints (15) ensure that the interval sequence variable V_d which consists of optional interval variables constitutes the order of the non-overlapping intervals for each machine $d \in D$. Moreover, it also has *TransitionDistance* function ($TD(t_{jj'})$) which puts a minimal time ($t_{jj'}$) to be maintained between the end of interval variable Y_{jd} and the start of interval variable $Y_{j'd}$.

2.3. Hybrid Modeling and Decomposition

A powerful aspect of ILP techniques is that the impacts of all constraints are evaluated simultaneously, and therefore it has a *global perspective* while the search tree is being explored (Rodosek et al. (1999); Harjunkoski and Grossmann (2002)). On the other hand, CP propagation algorithms explore the impacts of constraints sequentially through domain reduction of variables (*local perspective*) (Jain and Grossmann (2001)). These two important features, *global vs. local perspectives*, originate from the unique differences in defining models (constraints and variables) within these two techniques. These differences have significant impact on the subsequent search procedures. The most important bottleneck arises when the integrality constraints are tackled in the branch-and-bound search tree of the ILP. This process might result in evaluating an exponential number of combinations due to the number of subproblems. Furthermore, if the initial gap between the objective value for the optimal solution and the initial relaxed linear subproblem is large, the effectiveness of ILP tends to degrade. In addition to the number of the constraints and variables, representation of complex relationships between/among variables and constraints can be a major difficulty in providing a concise model. This is because, ILP can handle only inequality and equality constraints which might be insufficient to represent real life constraints. On the other hand, such a strong limitation on constraint expression is minimized in CP since application-based global constraints can be utilized to express complex relationships with no need for reformulation. However, selecting the most appropriate global constraint is crucial since the quality and speed of the domain reduction process at each node depends on the filtering algorithms running behind these global constraints. Since these propagation algorithms are called multiple times during a search, inefficient algorithms might dramatically slow down the search process. It should be remembered that not all global constraints have efficient constraint propagation

engines (Jain and Grossmann (2001)). Therefore, hybrid approaches aim to develop integrated methods to merge the complementary strengths of MP and CP to solve problems that are intractable using either of these two methods alone. Modeling an entire problem in both the CP and ILP contexts is referred to as *double modeling*.

More recently, Benders decomposition and Branch & Price algorithms have been very effective when reformulated in a hybrid CP and ILP framework. In the Branch & Price algorithm, the master problem and subproblem are formulated in ILP and CP, respectively (Topaloglu and Ozkarahan (2011); He and Qu (2012)). Using CP as a column generator takes advantage of CP's flexibility to formulate complex relationships that might occur in pricing problems (Hooker (2006a)). Similarly, the classic Benders decomposition master problem is formulated as an ILP and resolved with the Benders cuts generated from the CP formulated subproblems (Hooker (2006a, 2007b); Jain and Grossmann (2001)).

2.3.1. Benders decomposition: IP/CP integration

We realize that (CP-DS) model provides a feasible solution for all of the problem instances introduced in Section 3. However, it fails to report the optimal solution within a specified time limit. In order to overcome this weakness, we propose two novel logic-based Benders decomposition algorithms based on (DS) and (CP-DS). As a first step towards the optimality, we utilize the solution generated by the (CP-DS) model. We define z^{CP} as the objective function value of the best solution found by the (CP-DS) model within 300 seconds which also serves as a lower bound for the entire problem.

Hooker (2007b) states that classical Benders decomposition is not appropriate for highly combinatorial problems such as scheduling, because it enforces subproblems to be continuous linear or nonlinear programming problems. Therefore, recent studies have focused on implementing logic-based Benders decomposition in which subproblems are discrete feasibility problems and solved to generate Benders cuts. After the master problem is solved to optimality, all subproblems are solved and feasibility cuts are added to the master problem. If the master problem is solved to optimality and all subproblems are feasible, the solution is optimal to the global problem. However, if there is at least one infeasible subproblem, corresponding cuts are added to the cut set, and the master problem is called to perform the next iteration of the logic-based Benders algorithm. At a specific iteration (k) in which an infeasible solution is obtained by the subproblem(s), we keep the objective function value generated by the feasible master problem and use it as an upper bound throughout the remaining iterations. Therefore, we define z_k^{B1} and z_k^{B2} as the objective function values obtained by the master problems of Benders 1 and 2, respectively, at iteration k.

Given the competitive advantage of ILP in proving optimality through linear relaxation, the objective function of the problem is modeled in the master problem of logic-based Benders decomposition (M1-DS) which contains assignment (16) and budget (17) constraints. Lower and upper bound restrictions on the objective function are enforced by constraints (19) and (20). In order to limit the symmetric solutions and tighten the solution space, additional inequalities are added to (M1-DS). Constraints (21) make sure that total processing time of machine d cannot exceed

the planning horizon length ($|T|$). Finally, constraints (22) prevent assigning job j to machine d if the sum of the total restricted period length of job j and its processing time on d exceeds $|T|$.

$$\text{maximize} \sum_{j \in J} q_j \left(\sum_{d \in D} y_{dj} \right)$$

subject to (M1-DS)

$$\sum_{d \in D} y_{dj} \leq 1 \qquad j \in J \tag{16}$$

$$\sum_{j \in J} c_j \left(\sum_{d \in D} y_{dj} \right) \leq B \tag{17}$$

$$\sum_{j \in H_d^k} y_{d'j} \leq |H_d^k| - 1 \qquad d' \in \bar{D}_d;\ k = \{1, 2, \ldots, K-1\} \tag{18}$$

$$\sum_{j \in J} q_j \left(\sum_{d \in D} y_{dj} \right) \geq z^{CP} \tag{19}$$

$$\sum_{j \in J} q_j \left(\sum_{d \in D} y_{dj} \right) \leq \min_{k=1,2,\ldots,K-1} \{z_k^{B1}\} \tag{20}$$

$$\sum_{j \in J} p_{jd} y_{dj} \leq |T| \qquad d \in D \tag{21}$$

$$y_{dj} \left(p_{jd} + \sum_{w \in W_j} (e_w - b_w) \right) \leq |T| \qquad d \in D;\ j \in J \tag{22}$$

$$y_{dj} \in \{0, 1\} \qquad d \in D;\ j \in J$$

Note that (M1-DS) has no impact on the scheduling and sequencing decisions of jobs with respect to the machine they are assigned to. Therefore, we need to solve $|D|$ independent subproblems to check if the assignments made by (M1-DS) are feasible or not. Hence, let $H_d^k = \{j | y_{dj}^k = 1\}$ be the set of jobs $j \in J$ that are assigned to machine $d \in D$ at k^{th} iteration. We next define A_{jd} and $\bar{A}_d$ where A_{jd} is the **compulsory (not optional)** interval variable associated with job j and machine d and $\bar{A}_d$ is the**interval sequence variable** for machine $d \in D$. Then, the following *feasibility problem* (without an objective function) is formulated in the CP context and solved for each machine d in order to make specific scheduling and sequencing decisions subject to restricted periods and travel times between jobs.

(S1-DS: Subproblem for each $d \in D$)

$$NoOverlap(\bar{A}_d, t_{jj'})$$

$$A_{jd}.StartMin = 1 \qquad j \in H_d^k$$

$$A_{jd}.EndMax = |T| \qquad j \in H_d^k$$

$$ForbidExtent(A_{jd}, I(j)) \qquad j \in H_d^k$$

$$Cumulative(A_{jd}, 1, 1)$$

If (S1-DS) is feasible for each machine, the solution for both master and subproblems will be the optimal solution to the overall problem. Otherwise, a "Benders cut" will be generated for each machine that fails to provide a feasible schedule for the assigned jobs. Such cuts are first offered by Hooker et al. (1999) and referred to as "no good" and later used by Jain and Grossmann (2001) on a parallel machine scheduling problem. These studies report a significant decrease in solution times when logic based Benders decomposition is applied with "no good" compared to pure CP or ILP formulations. Assume that jobs in H_d^k are not successfully completed by machine d. Also, let $\bar{D}_d = \{d' | r_{d'} \leq r_d,\ d, d' \in D, d \neq d'\}$ be the set of machine operation rates which are less than or equal to the operation rate of $d \in D$. Then, following cuts are formed;

$$\sum_{j \in H_d^k} y_{dj} \leq |H_d^k| - 1 \tag{23}$$

$$\sum_{j \in H_d^k} y_{d'j} \leq |H_d^k| - 1 \qquad d' \in \bar{D}_d. \tag{24}$$

Note that "no good" (23) can be satisfied by omitting just one job from the set H_d^k in later iterations. Moreover, constraints (23) cannot prevent assigning same jobs in H_d^k to another machine $d' \in \bar{D}_d$. In order to prevent this, we developed the cuts in (24). Thus, if the subproblem for machine d at iteration k is infeasible, then inequalities (24) are added to (M1-DS) to enforce that all jobs in H_d^k cannot be assigned to machines $d' \in \bar{D}_d$ for the later iterations. Note that such inequalities cut off several assignment combinations which might only be revealed by several branching & propagation operations in a CP search when the whole problem is approached by a pure CP formulation. Moreover, since all the subproblems are solved independently in this decomposition approach, it is easy to identify which subproblem (machine) produces infeasible solutions. The size of the master problem increases as the the algorithm generates cuts. However, the size of each subproblem cannot exceed the total number of jobs, $|J|$. As a consequence, if the optimal solution is not obtained in the early iterations of the algorithm, the time required for solving (M1-DS) to optimality will increase significantly. In order to lessen the intensity of the Benders cut added to (M1-DS), we developed a second logic-based Benders decomposition algorithm. The master and subproblem of this alternative approach are given below as (M2-DS) and (S2-DS), respectively.

$$\text{maximize} \sum_{j \in J} q_j x_j$$

subject to (M2-DS)

$$\sum_{j\in J} c_j x_j \leq B \tag{25}$$

$$\sum_{j\in G^k} x_j \leq |G^k| - 1 \qquad k = \{1, 2, \ldots, K-1\} \tag{26}$$

$$\sum_{j\in J} q_j x_j \geq z^{CP} \tag{27}$$

$$\sum_{j\in J} q_j x_j \leq \min_{k=1,2,\ldots,K-1} \{z_k^{B2}\} \tag{28}$$

$$x_j \left(\min_{d\in D} \{p_{jd}\} + \sum_{w\in W_j} (e_w - b_w) \right) \leq |T| \qquad j \in J \tag{29}$$

$$x_j \in \{0, 1\} \qquad j \in J$$

where x_j is a decision variable and will be 1 if job j is completed by any machine, 0 otherwise. $G^k = \{j | x_j^k = 1\}$ is the set of jobs that are decided to be processed at iteration k. The objective of M2-DS is to maximize the total profit subject to budget constraint (25), Benders cuts (26) and some additional inequalities in order to cut off infeasible or low-quality solutions in the early stages of the algorithm. The lower and upper bound restrictions are enforced by constraints (27) and 28, respectively. Constraints (29) ensure that the sum of the minimum processing time and total restricted period length of job j does not exceed the planning horizon length ($|T|$). Since the job-machine assignment decisions are not made by (M2-DS), they have to be handled in (S2-DS) through *optional interval variables*. Hence, let Y_{jd} be the optional interval variable when job $j \in G^k$ is assigned to machine $d \in D$ at iteration k and V_d be the interval sequence variable for machine $d \in D$. Therefore, we can write the subproblem as

(S2-DS)

$$Alternaive(Z_j, Q_j) \qquad j \in G^k \tag{30}$$

$$Z_j.StartMin = 1 \qquad j \in G^k \tag{31}$$

$$Z_j.EndMax = |T| \qquad j \in G^k \tag{32}$$

$$ForbidExtent(Z_j, I(j)) \qquad j \in G^k \tag{33}$$

$$NoOverlap(V_d, t_{jj'}) \qquad d \in D \tag{34}$$

$$Cumulative(Z_j, 1, |D|) \qquad j \in G^k \tag{35}$$

where Z_j is the compulsory interval variable for job $j \in G^k$ and $Q_j = \{Y_{j1}, Y_{j2}, \ldots, Y_{j|D|}\}$ is the set of interval variables Y_{jd} for each $j \in G^k$. Similar to problem (CP-DS), (S2-DS) controls the job assignments to machines. However, in this case, Z_j is a compulsory interval variable that must be present in the solution because M2-DS determines the set of jobs that must be processed. Therefore, constraints (30) assure that each job $j \in G^k$ is processed

by exactly one machine. Constraints (31) and (32) enforce that each job must be started and finished within the given planning horizon. Moreover, constraints (33) forbid the restricted periods of each job $j \in G^k$, represented by intensity function ($I(j)$), overlapping with its processing time. Similarly, constraints (34) make sure that machine d cannot operate while traveling between jobs j and j' such that $j, j' \in G^k$ and $j \neq j'$. Finally, (35) is a redundant constraint that strengthens the formulation by ensuring that the total number of machines in operation cannot exceed the fleet size at any given time. Note that there are only two problems that need to be solved at each iteration in the second Benders decomposition algorithm. However, the size of subproblem (S2-DS) is significantly larger than the ones in (S1-DS). Performance of these two algorithms are evaluated in Section 4.

3. Case Study: Optimizing Inland Waterway Infrastructure Maintenance for Supply Chain Operations

Each year the U.S. Army Corps of Engineers (USACE) dredges hundreds of navigation projects through its fleet of government dredges and individual contracts with private industry. The decision of assigning dredge resources (government and private industry) to navigation projects is predominately made regionally by awarding the contract to the lowest cost bid that meets the scheduling demands of the dredge job. Most likely, efficiencies can be gained by optimizing the entire portfolio of dredging jobs. The proposed models and solution approaches in Section 2 are used to optimize the decision of allocating dredge resources to projects under necessary constraints such as environmental windows, dredge resource cost and availability, and sequence dependent travel times. Using these approaches, sensitivity analysis on resource levels are performed in order to demonstrate under which circumstances USACE can complete the entire dredging portfolio while achieving compliance and desired system performance.

A specific challenge investigated in this study is the concept of environmental window restrictions. The USACE describes environmental windows as temporal constraints placed upon dredged material disposal operations in order to protect biological resources or their habitats from potentially detrimental effects (Dickerson et al. (I998)). The USACE has documented an increase in total dredging cost without a proportionate increase in total volume of material dredged (Pointon (1996)). Dickerson et al. (I998) stated that a widely-held explanation for this increase in dredging costs is system inefficiencies associated with environmental window compliance. In order to be compatible with the related model constraints, we define restricted periods for each environment window (job availability interval).

Dredge fleet scheduling and sequencing optimization is challenging due to the highly variable and uncertain feature of natural processes, engineering capacity, dredging operations and economic conditions (Ratick and Garriga (1996); Gedik (2014)). Despite its difficulty level, risk and reliability based dredging optimization papers (see Ratick and Garriga (1996); Menon and Lansey (1990); Lund (1990); Nachtmann et al. (2014)) are frequently seen in the literature that model the volatile river and environmental situations that influence dredging operations at a specific project or reach level. The solution techniques in this study provide a system wide optimization perspective which can efficiently and effectively use resources across the entire dredging project portfolio.

Historical USACE dredge project data collected between 1997 and 2011 was utilized to parameterize the model. The data was provided by the Corps Dredging Information System, and a total of 116 unique channel maintenance dredging jobs were identified as seen in Figure 2. Table 2 demonstrates the descriptive statistics of the model input parameters such as volume of jobs (q_j), cost of jobs (c_j), length of restricted periods ($e_w - b_w$) and production rate of dredge vessels (r_d). Since the USACE cannot currently afford to meet all the dredging requests, we set the available budget (B) to 75% of the total cost of the 116 jobs. A from-to distance matrix was constructed by using a GIS layer to compute travel distance on the waterways between prospective dredge project locations. Then, the travel (setup) time between each job pair is calculated by assuming the velocity of a dredge vessel is 50 miles per day.

Table 2: Descriptive Statistics of Input Parameters

	Average	Minimum	Maximum	St. Dev.
q_j (cubic yards)	416427.4	4376.4	5413965.0	705142.0
c_j	$1,922,517.34	$46,440.77	$14,477,345.28	$2,455,009.14
$e_w - b_w$ (days)	144.6	30.0	275.0	71.5
r_d (cubic yards per day)	14636.7	1237.7	66418.0	14584.4

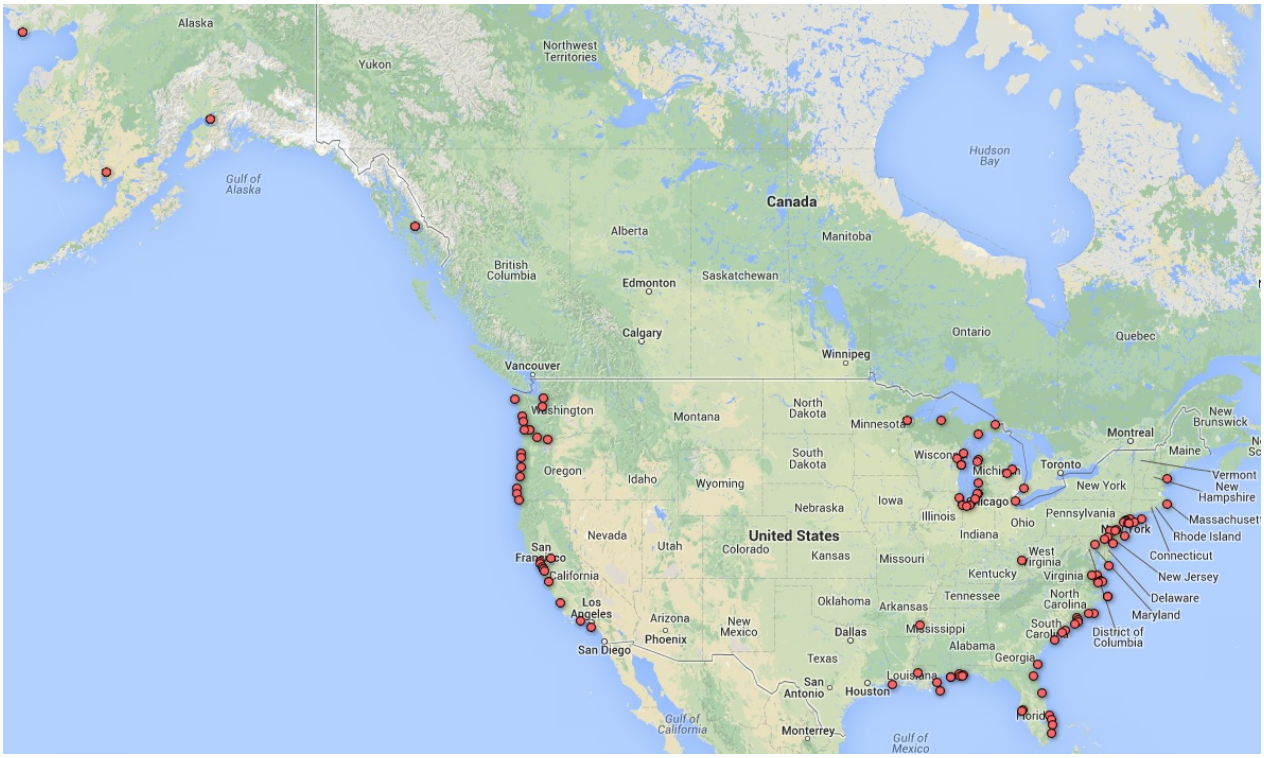

Figure 2: Graphical Depiction of 116 Dredge Project Locations

We identified 130 distinct restricted periods across the 116 jobs that are time intervals when dredging is not permitted at certain job locations due to environmental concerns. Note that some jobs may have none, one or multiple

restricted periods. Production rates of dredge vessels in Table 2 reflect a historical statistical average of multiple dredging projects.

Table 3: Experiment Design

$\|J\|$	Number of RPs	$\|D\|$	Job Size (q_j)	Vessel Type
32	33	10, 15, 20, 25, 30	Original (O) Random (R)	Fastest (F) Random (R) Slowest (S)
57	63	10, 15, 20, 25, 30	Original (O) Random (R)	Fastest (F) Random (R) Slowest (S)
116	127	10, 15, 20, 25, 30	Original (O) Random (R)	Fastest (F) Random (R) Slowest (S)

Table 3 demonstrates the properties of 90 problem instances that were used to measure the performance of the solution techniques described in Section 2. The first ($|J| = 32$) and second ($|J| = 57$) sets of jobs do not have any common dredge projects; whereas, the final job set ($|J| = 116$) includes all job locations and restricted periods. For each problem instance, we either select the fastest, slowest or randomly picked distinct dredge vessels to build the fleet from a 40 vessel fleet. We also generate job sizes while keeping all other parameters constant with respect to a discrete uniform distribution in a range of $[10K, 1M]$. Thus, "32-10-33-O-F" stands for the problem instance with 32 jobs with Original job sizes, 33 restricted periods and the fastest 10 dredge vessels of the fleet.

4. Computational Results

The parallel machine scheduling problem formulations in Section 2 are modeled in IBM ILOG CPLEX Optimization Studio 12.6 (IBM (2014)) which uses IBM ILOG CPLEX 12.6 to solve ILP, IBM ILOG CP Optimizer 12.6 to solve CP and both to solve logic based Benders algorithms. All formulations are modeled in C++ programming language. IBM ILOG Concert Technology is utilized for embedding the formulations in C++ language into IBM ILOG CPLEX and CP Optimizer. We run all test problems on a Core 2 Duo 2.93 GHz, 16 GB RAM computer. We create and solve the master problems from scratch with the updated Benders cuts at each iteration.

As mentioned before, we observe that for a medium size problem instance ($|D| = 10$ and $|J| = 32$), CPLEX cannot even begin to solve the (DS) model presented in Section 2.1 due to the large number of decision variables and constraints. Therefore, we do not report any experiment results for models (DS). Table 4, 5 and 6 demonstrate the performance of the Benders decomposition algorithms over the problem instances with $|J| = 32$, 57 and 116, respectively. These results are obtained by solving the parallel machine scheduling problem with maximizing total profit objective. For each algorithm, we report total run time, total master and subproblem(s) run time, number of iterations and final gap on each problem instance. Initial 300 seconds (CP-DS) model run time to get a lower

bound (z^{CP}) is excluded from the time statistics listed in these tables. Finally, Table 7 comparatively illustrate the number of optimal solutions obtained by the decomposition algorithms for the problem instances with original and randomly generated job sizes. Unless an optimal solution is attained, Benders 1 and 2 terminate the search under three circumstances:

(i) the total algorithm time reaches 10,800 seconds or,

(ii) the number of iterations is equal to 2,000 or,

(iii) $\epsilon = \frac{z_k^{B2} - z^{CP}}{z_k^{B2}} \leq 0.01$ and $\epsilon = \frac{z_k^{B1} - z^{CP}}{z_k^{B1}} \leq 0.01$ at iteration k.

Decomposition algorithms are very efficient and effective in providing $\epsilon-$optimal solutions within a reasonable amount of time and allowed iteration limit. Table 7 points out that out of 90 problem instances, 75 and 83 $\epsilon-$optimal solutions are achieved by Benders 1 and 2, respectively. It also shows that 39 (44) of the 45 problem instances with original job sizes and 36 (38) out of 45 problem instances with randomly generated job sizes are solved to near optimality by Benders 1 (Benders 2).

Both algorithms are able to capture a feasible solution for the problem instances with no optimal solution (highlighted in *italic* font in Table 4, 5 and 6). Larger solution times and number of iterations are required for both Benders 1 and 2 as the number of jobs increases. However, the average master and subproblem solution times of Benders 2 are smaller than counterparts produced by Benders 1 although Benders 2 requires significantly more number of iterations, on average. Note that assignment of jobs to machines are handled by the master problem of Benders 1 whereas subproblem of Benders 2 makes the same decisions once its master problem identifies the set of jobs need to be processed at each iteration. Moreover, feasibility cuts generated by Benders 1 after an infeasible subproblem associated with machine d are also applied to other machines with smaller operation rates. Hence, Benders 1 adds more feasibility cuts to its master problem than Benders 2 at a given iteration and as the algorithm progresses, solving the master problem of Benders 1 gets more difficult and requires more time than the one in Benders 2. Despite this cost, the feasibility cuts of the first variant carry more information from subproblems to master problem and therefore, eliminate more inferior solutions at an iteration which ultimately results in exploring near optimal solutions at earlier iterations. On the other hand, it is faster to solve a single feasibility subproblem in the second variant even with assignment decisions. This situation exemplifies the high efficiency of CP in obtaining feasible/infeasible solutions for highly constrained problems. By exploiting the competitive advantage of CP in delivering quick feasible solutions, Benders 2 evaluates many more different job combinations in more iterations. As a consequence, it outperforms the first variant in terms of number of near optimal solutions across all problem instances.

In order to assess the quality of the solutions obtained by the (CP-DS) model with 300 and 10800 seconds time limit, we calculate the optimality gap % based on the optimal solutions obtained by the decomposition algorithms for the problem with maximizing total profit objective as shown below.

$$\text{Optimality Gap } \% \ = 100 * \left(\frac{\text{Optimal Objective Value} - \text{(CP-DS) Objective Value}}{\text{Optimal Objective Value}} \right)$$

Table 8 shows the optimality gap % of the (CP-DS) model for the 86 problem instances that are solved to optimality by at least one of the decomposition algorithms. An interesting observation is that although the (CP-DS) model with 300 and 10800 seconds time limit terminates with an optimal solution for only eight problem instances, the average optimality gap is 1.12% and 0.77%, respectively. This indicates that the pure CP model is able to explore high quality solutions (optimal with 0% optimality gap for some instances) early, but it is only able to prove the optimality of eight of them. These findings also suggest that the lower bound (z^{CP}) obtained from the (CP-DS) model within 300 seconds is very useful in finding $\epsilon-$optimal solutions by Benders 1 and 2 algorithms.

Table 4: Comparison of Benders 1 & 2 algorithms on problem instances with $|J| = 32$

Problem	Benders 1 (M1-DS, S1-DS)					Benders 2 (M2-DS, S2-DS)				
	Total Time	M1-DS Time	S1-DS Time	Iteration	ϵ	Total Time	M2-DS Time	S2-DS Time	Iteration	ϵ
32-10-33-O-F	1.25	1.04	0.21	1	0.00	0.76	0.33	0.43	1	0.00
32-10-33-O-R	1380.63	396.18	984.46	459	0.00	682.35	497.48	184.87	1096	0.00
32-10-33-O-S	260.22	53.42	206.80	74	0.00	697.48	510.88	186.60	993	0.01
32-10-33-R-F	66.15	15.54	50.61	25	0.00	0.71	0.31	0.40	1	0.00
32-10-33-R-R	3.98	1.09	2.89	1	0.00	0.74	0.18	0.56	1	0.00
32-10-33-R-S	32.55	6.53	26.02	11	0.00	61.23	42.44	18.79	115	0.01
32-15-33-O-F	1.83	1.59	0.25	1	0.00	0.86	0.37	0.49	1	0.00
32-15-33-O-R	1.80	1.55	0.24	1	0.01	0.71	0.34	0.37	1	0.01
32-15-33-O-S	71.49	12.67	58.81	15	0.00	117.72	81.73	35.99	161	0.01
32-15-33-R-F	1.49	1.15	0.34	1	0.00	0.84	0.19	0.66	1	0.00
32-15-33-R-R	92.89	14.88	78.01	23	0.00	0.89	0.35	0.53	1	0.00
32-15-33-R-S	236.37	48.01	188.36	63	0.00	256.94	182.21	74.73	476	0.01
32-20-33-O-F	2.23	1.90	0.33	1	0.00	0.97	0.44	0.54	1	0.00
32-20-33-O-R	87.85	24.94	62.91	31	0.00	0.91	0.37	0.54	1	0.00
32-20-33-O-S	4.18	1.37	2.81	1	0.00	0.83	0.22	0.61	1	0.00
32-20-33-R-F	1.22	0.98	0.25	1	0.00	0.65	0.15	0.50	1	0.00
32-20-33-R-R	350.86	84.53	266.33	113	0.00	1.04	0.43	0.61	1	0.00
32-20-33-R-S	*10800.00*	*10410.02*	*389.98*	*96*	*0.18*	766.08	545.48	220.60	816	0.01
32-25-33-O-F	31.32	10.73	20.59	7	0.00	0.76	0.38	0.38	1	0.00
32-25-33-O-R	4.98	4.15	0.83	1	0.01	2.02	1.07	0.95	1	0.01
32-25-33-O-S	12.46	3.76	8.70	1	0.00	1.51	0.44	1.07	1	0.00
32-25-33-R-F	3.37	2.34	1.03	1	0.00	1.23	0.35	0.87	1	0.00
32-25-33-R-R	95.35	20.20	75.15	17	0.00	1.50	0.72	0.79	1	0.00
32-25-33-R-S	12.29	3.48	8.81	1	0.00	1.44	0.36	1.08	1	0.00
32-30-33-O-F	3.09	2.59	0.49	1	0.00	1.33	0.68	0.65	1	0.00
32-30-33-O-R	3.62	2.95	0.67	1	0.00	1.36	0.68	0.68	1	0.00
32-30-33-O-S	50.98	10.82	40.16	5	0.00	1982.79	1438.44	544.35	1356	0.01
32-30-33-R-F	129.95	24.23	105.71	16	0.00	1.32	0.57	0.75	1	0.00
32-30-33-R-R	3.53	2.86	0.68	1	0.00	1.41	0.41	0.99	1	0.00
32-30-33-R-S	9.65	2.43	7.22	1	0.00	1.11	0.33	0.79	1	0.00
Average	458.59	372.26	86.32	32.37	0.01	152.98	110.28	42.71	167.87	0.00

Table 5: Comparison of Benders 1 & 2 algorithms on problem instances with $|J| = 57$

Problem	Benders 1 (M1-DS, S1-DS)					Benders 2 (M2-DS, S2-DS)				
	Total Time	M1-DS Time	S1-DS Time	Iteration	ϵ	Total Time	M2-DS Time	S2-DS Time	Iteration	ϵ
57-10-63-O-F	3.17	2.79	0.38	1	0.00	1.90	0.82	1.07	1	0.00
57-10-63-O-R	5.61	2.95	2.66	1	0.00	2.04	0.93	1.11	1	0.00
57-10-63-O-S	*10800.00*	*8051.93*	*2748.07*	*813*	*0.03*	715.43	529.10	186.33	643	0.01
57-10-63-R-F	6.28	3.15	3.13	1	0.00	2.25	1.00	1.25	1	0.00
57-10-63-R-R	*10800.00*	*7390.42*	*3409.58*	*775*	*0.02*	2.62	1.11	1.51	1	0.00
57-10-63-R-S	1801.51	1216.60	584.91	121	0.01	2062.34	1553.49	508.85	1454	0.01
57-15-63-O-F	3.41	2.96	0.45	1	0.00	1.66	0.91	0.75	1	0.00
57-15-63-O-R	3.61	3.16	0.46	1	0.00	2.14	0.93	1.21	1	0.00
57-15-63-O-S	363.75	98.15	265.60	50	0.01	937.18	689.23	247.96	820	0.01
57-15-63-R-F	9.57	4.36	5.21	1	0.00	2.18	0.83	1.35	1	0.00
57-15-63-R-R	3706.28	1285.22	2421.06	338	0.01	2.79	1.12	1.68	1	0.00
57-15-63-R-S	*10800.00*	*10794.55*	*5.45*	*1*	*0.15*	2552.63	1942.09	610.53	1665	0.01
57-20-63-O-F	5.08	3.48	1.60	1	0.00	2.00	0.91	1.09	1	0.00
57-20-63-O-R	5.58	4.79	0.78	1	0.00	2.12	0.93	1.19	1	0.00
57-20-63-O-S	34.03	9.36	24.67	3	0.00	438.41	316.46	121.95	328	0.01
57-20-63-R-F	14.71	7.84	6.88	1	0.00	3.76	1.30	2.46	1	0.00
57-20-63-R-R	13.24	5.53	7.71	1	0.00	2.02	0.75	1.27	1	0.00
57-20-63-R-S	*10800.00*	*10558.22*	*241.79*	*42*	*0.17*	657.43	474.32	183.11	646	0.01
57-25-63-O-F	7.20	3.52	3.68	1	0.00	1.81	0.72	1.09	1	0.00
57-25-63-O-R	6.22	4.25	1.97	1	0.00	1.85	0.84	1.01	1	0.00
57-25-63-O-S	170.75	38.49	132.26	17	0.00	724.00	525.80	198.19	668	0.01
57-25-63-R-F	12.10	5.20	6.90	1	0.00	2.54	0.89	1.65	1	0.00
57-25-63-R-R	850.06	196.38	653.67	82	0.00	2.19	0.76	1.44	1	0.00
57-25-63-R-S	*10800.00*	*10319.30*	*480.70*	*48*	*0.03*	47.41	33.03	14.38	37	0.01
57-30-63-O-F	15.78	6.62	9.16	1	0.00	2.30	0.99	1.31	1	0.00
57-30-63-O-R	9.67	5.06	4.60	1	0.00	1.74	0.72	1.02	1	0.00
57-30-63-O-S	936.97	206.21	730.75	64	0.00	2.59	1.12	1.47	1	0.00
57-30-63-R-F	12.44	5.34	7.10	1	0.00	2.51	0.81	1.70	1	0.00
57-30-63-R-R	15.75	6.58	9.17	1	0.00	2.88	1.11	1.77	1	0.00
57-30-63-R-S	14.87	5.28	9.59	1	0.00	2.72	1.01	1.71	1	0.00
Average	2067.59	1674.92	392.66	79.07	0.02	272.85	202.80	70.05	209.43	0.00

Table 6: Comparison of Benders 1 & 2 algorithms on problem instances with $|J| = 116$

Problem	Benders 1 (M1-DS, S1-DS)					Benders 2 (M2-DS, S2-DS)				
	Total Time	M1-DS Time	S1-DS Time	Iteration	ϵ	Total Time	M2-DS Time	S2-DS Time	Iteration	ϵ
116-10-127-O-F	7.32	4.86	2.46	1	0.01	2.88	1.14	1.73	1	0.01
116-10-127-O-R	*10800.00*	*7287.51*	*3512.49*	*664*	*0.13*	5773.63	1776.71	3996.93	1291	0.01
116-10-127-O-S	8938.89	7445.27	1493.62	639	0.01	*2542.43*	*2189.41*	*353.03*	*2000*	*0.13*
116-10-127-R-F	3.54	2.15	1.39	1	0.01	2.80	0.53	2.27	1	0.01
116-10-127-R-R	*10800.00*	*9632.84*	*1167.16*	*674*	*0.10*	*10800.00*	*2.51*	*10797.49*	*3*	*0.28*
116-10-127-R-S	186.08	101.51	84.57	47	0.01	*2645.45*	*2244.70*	*400.75*	*2000*	*0.57*
116-15-127-O-F	69.99	27.65	42.34	11	0.01	2.50	0.77	1.72	1	0.00
116-15-127-O-R	180.72	72.50	108.22	35	0.01	1.93	0.65	1.28	1	0.00
116-15-127-O-S	*10800.00*	*8750.62*	*2049.38*	*385*	*0.08*	3061.62	2598.63	462.99	1856	0.01
116-15-127-R-F	14.62	8.21	6.41	1	0.00	5.47	1.57	3.89	1	0.00
116-15-127-R-R	2692.57	1518.06	1174.51	172	0.01	*10800.00*	*6.39*	*10793.61*	*3*	*0.04*
116-15-127-R-S	*10800.00*	*7799.39*	*3000.61*	*458*	*0.02*	*4914.96*	*4054.80*	*860.16*	*2000*	*0.55*
116-20-127-O-F	176.36	61.37	115.00	10	0.01	3.81	1.24	2.57	1	0.00
116-20-127-O-R	851.51	315.86	535.65	56	0.01	3.18	1.14	2.04	1	0.00
116-20-127-O-S	*10800.00*	*8780.46*	*2019.54*	*288*	*0.09*	1168.75	872.31	296.44	524	0.01
116-20-127-R-F	17.86	9.24	8.61	1	0.00	4.49	0.92	3.57	1	0.00
116-20-127-R-R	14.09	7.76	6.33	1	0.01	6.16	1.49	4.67	1	0.01
116-20-127-R-S	*10800.00*	*8614.18*	*2185.82*	*299*	*0.04*	*6044.62*	*4924.11*	*1120.51*	*2000*	*0.44*
116-25-127-O-F	357.20	111.37	245.83	18	0.01	6.13	1.31	4.82	1	0.00
116-25-127-O-R	1462.93	578.62	884.31	78	0.01	3.58	1.13	2.44	1	0.00
116-25-127-O-S	*10800.00*	*8057.15*	*2742.86*	*253*	*0.07*	1135.02	839.49	295.54	628	0.01
116-25-127-R-F	1077.77	364.48	713.30	54	0.01	6.51	1.30	5.21	1	0.00
116-25-127-R-R	16.96	9.08	7.88	1	0.00	4.70	1.21	3.48	1	0.00
116-25-127-R-S	*10800.00*	*10096.17*	*703.83*	*72*	*0.11*	*10800.00*	*1597.98*	*9202.02*	*894*	*0.20*
116-30-127-O-F	402.31	125.30	277.01	21	0.01	3.83	1.26	2.57	1	0.00
116-30-127-O-R	626.34	205.91	420.43	34	0.01	3.81	1.24	2.58	1	0.00
116-30-127-O-S	*10800.00*	*6640.16*	*4159.84*	*199*	*0.02*	4417.12	3335.16	1081.97	1088	0.01
116-30-127-R-F	1906.27	713.79	1192.48	64	0.01	8.36	2.19	6.17	1	0.00
116-30-127-R-R	1072.53	425.73	646.80	40	0.01	3.82	1.19	2.62	1	0.00
116-30-127-R-S	8601.94	6719.69	1882.25	176	0.01	*10800.00*	*4.48*	*10795.52*	*3*	*0.04*
Average	4195.93	3149.56	1046.36	158.43	0.03	2499.25	815.57	1683.69	476.90	0.08

Table 7: Number of $\epsilon-$optimal solutions found by Benders 1 and 2

		Benders 1		Benders 2	
$\|J\|$	Number of Instances	$q_j \sim$ Original	$q_j \sim DU(10K, 1M)$	$q_j \sim$ Original	$q_j \sim DU(10K, 1M)$
32	30	15	14	15	15
57	30	14	11	15	15
116	30	10	11	14	8
Total		39	36	44	38

Table 8: % Optimality gap of (CP-DS) model with 300 seconds and 10800 seconds time limit

Problem Instance	CP-DS (300 s.)	CP-DS (10800 s.)	Problem Instance	CP-DS (300 s.)	CP-DS (10800 s.)	Problem Instance	CP-DS (300 s.)	CP-DS (10800 s.)
32-10-33-O-F	0.00%	0.00%	57-10-63-O-F	0.47%	0.47%	116-10-127-O-F	0.55%	0.43%
32-10-33-O-R	3.57%	3.14%	57-10-63-O-R	0.05%	0.03%	116-10-127-O-R	0.97%	0.00%
32-10-33-O-S	0.97%	0.97%	57-10-63-O-S	0.97%	0.97%	116-10-127-O-S	0.98%	0.00%
32-10-33-R-F	2.30%	2.30%	57-10-63-R-F	0.00%	0.00%	116-10-127-R-F	0.61%	0.00%
32-10-33-R-R	0.00%	0.00%	57-10-63-R-R	7.51%	7.51%	116-10-127-R-S	0.96%	0.96%
32-10-33-R-S	0.94%	0.94%	57-10-63-R-S	0.97%	0.97%	116-15-127-O-F	1.28%	0.48%
32-15-33-O-F	0.00%	0.00%	57-15-63-O-F	0.47%	0.47%	116-15-127-O-R	1.66%	0.27%
32-15-33-O-R	0.58%	0.58%	57-15-63-O-R	0.47%	0.47%	116-15-127-O-S	0.95%	0.00%
32-15-33-O-S	0.96%	0.96%	57-15-63-O-S	1.00%	1.00%	116-15-127-R-F	0.32%	0.27%
32-15-33-R-F	0.00%	0.00%	57-15-63-R-F	0.00%	0.00%	116-15-127-R-R	0.99%	0.00%
32-15-33-R-R	2.30%	2.30%	57-15-63-R-R	1.01%	1.01%	116-20-127-O-F	1.17%	0.44%
32-15-33-R-S	0.93%	0.93%	57-15-63-R-S	14.91%	11.81%	116-20-127-O-R	2.23%	0.43%
32-20-33-O-F	0.00%	0.00%	57-20-63-O-F	0.40%	0.00%	116-20-127-O-S	1.00%	0.00%
32-20-33-O-R	1.41%	1.41%	57-20-63-O-R	0.47%	0.47%	116-20-127-R-F	0.32%	0.27%
32-20-33-O-S	0.00%	0.00%	57-20-63-O-S	0.98%	0.98%	116-20-127-R-R	0.91%	0.27%
32-20-33-R-F	0.00%	0.00%	57-20-63-R-F	0.00%	0.00%	116-25-127-O-F	1.55%	0.08%
32-20-33-R-R	2.30%	2.30%	57-20-63-R-R	0.00%	0.00%	116-25-127-O-R	2.33%	0.43%
32-20-33-R-S	0.98%	0.98%	57-20-63-R-S	1.00%	0.00%	116-25-127-O-S	0.95%	0.95%
32-25-33-O-F	1.70%	1.70%	57-25-63-O-F	0.07%	0.03%	116-25-127-R-F	2.36%	0.27%
32-25-33-O-R	0.69%	0.69%	57-25-63-O-R	0.47%	0.47%	116-25-127-R-R	0.27%	0.27%
32-25-33-O-S	0.00%	0.00%	57-25-63-O-S	0.99%	0.99%	116-30-127-O-F	1.62%	0.50%
32-25-33-R-F	0.00%	0.00%	57-25-63-R-F	0.00%	0.00%	116-30-127-O-R	1.84%	0.44%
32-25-33-R-R	2.50%	2.50%	57-25-63-R-R	2.13%	2.13%	116-30-127-O-S	0.99%	0.00%
32-25-33-R-S	0.00%	0.00%	57-25-63-R-S	0.98%	0.98%	116-30-127-R-F	2.36%	0.27%
32-30-33-O-F	0.00%	0.00%	57-30-63-O-F	0.49%	0.49%	116-30-127-R-R	2.09%	0.27%
32-30-33-O-R	0.00%	0.00%	57-30-63-O-R	0.47%	0.47%	116-30-127-R-S	0.94%	0.00%
32-30-33-O-S	0.94%	0.94%	57-30-63-O-S	2.69%	2.40%			
32-30-33-R-F	2.50%	2.50%	57-30-63-R-F	0.00%	0.00%			
32-30-33-R-R	0.00%	0.00%	57-30-63-R-R	0.00%	0.00%			
32-30-33-R-S	0.00%	0.00%	57-30-63-R-S	0.00%	0.00%			

5. Concluding Remarks

In this study, we discuss the advantages and disadvantages of three modeling approaches for the parallel machine scheduling problem with sequence dependent setup times, job availability intervals, unrelated machines and non-identical job durations. We propose three different problem formulations (i) integer linear programming, (ii) constraint programming, and (iii) logic-based Benders decomposition algorithms to tackle this problem with a *maximizing total profit* objective function.

We prepare real-life test instances in collaboration with the USACE in order to optimize their inland waterway infrastructure maintenance operations and make the waterways navigable for the supply chain activities. For most of the problem instances, we observe that the two proposed Benders algorithms are much more efficient and effective than the standalone ILP and CP models. Collectively, Benders 1 and 2 are able to obtain 86 $\epsilon-$optimal solutions out of 90 problem instances with original and randomly generated job sizes. Based on our further computational experiments, we observe that the number of optimal solutions for the 45 problem instances with randomly generated job sizes ($q_j \sim DU(10K, 2M)$) decreases to 22 whereas the number of optimal solutions for the problem instances with the original job sizes remains at the same level.

This work can be extended in several directions. First of all, this study assumes that the cost and profit of processing a job is independent from the machine type. This assumption can easily be relaxed in pure ILP and CP formulations. However, developing a decomposition algorithm with this extension requires further analysis. Second of all, soft transition periods between the time availability constraints and environmental work windows should be incorporated in the modeling. This can be easily made possible in CP by assigning different values to the intensity function where the operation rate is defined for specific time intervals. Finally, the master problems of the decomposition algorithms are created from scratch at every iteration in the current implementation scheme. A more efficient strategy would be to store the search trees of the master problems and tighten the formulation by adding the Benders cuts at each iteration. This approach can be implemented in several ways. However, the most recent and promising way is to use *lazy constraint* technology. In this method, Benders cuts would be treated as lazy constraints for which a callback routine is triggered every time an incumbent solution candidate is reached. Most of the lazy constraints are not expected to be active at the optimal solution. The callback will either result in validating the incumbent solution or rejecting it due to a violated Benders cut (lazy constraint). By this way, some time savings might be obtained since the search tree would not be created from scratch every time the master problem is invoked.

Acknowledgment

This material is based upon work supported by the U.S. Army Corps of Engineers under Contract Awards Number *W911HZ-12-P-0172*. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Army Corps of

Engineers.

References

Arkin, E. M. and Silverberg, E. B. (1987). Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics*, 18(1):1 – 8.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252.

Cakici, E. and Mason, S. J. (2007). Parallel machine scheduling subject to auxiliary resource constraints. *Production Planning and Control*, 18(3):217–225.

Cesta, A., Oddi, A., and Smith, S. (2002). A constraint-based method for project scheduling with time windows. *Journal of Heuristics*, 8:109–136.

Dickerson, D., Reine, K. J., and Clarke, D. G. (I998). Economic impacts of environmental windows associated with dredging operations. Technical report, U.S. Army Engineer Research and Development Center, Vicksburg, MS, www.wes.army.mil/el/dots/doer.

Edis, E. B. and Oguz, C. (2012). Parallel machine scheduling with flexible resources. *Computers & Industrial Engineering*, 63(2):433 – 447.

Edis, E. B. and Ozkarahan, I. (2011). A combined integer/constraint programming approach to a resource-constrained parallel machine scheduling problem with machine eligibility restrictions. *Engineering Optimization*, 43(2):135–157.

Focacci, F., Lodi, A., and Milano, M. (2002). Mathematical programming techniques in constraint programming: A short overview. *Journal of Heuristics*, 8:7–17.

Gedik, R. (2014). *Large-Scale Solution Approaches for Healthcare and Supply Chain Scheduling*. PhD thesis, UNIVERSITY OF ARKANSAS.

Harjunkoski, I. and Grossmann, I. E. (2002). Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Comp. Chem. Engng*, 26:1533–1552.

He, F. and Qu, R. (2012). A constraint programming based column generation approach to nurse rostering problems. *Computers & Operations Research*, 39(12):3331 – 3343.

Heipcke, S. (1999). Comparing constraint programming and mathematical programming approaches to discrete optimisation-the change problem. *The Journal of the Operational Research Society*, 50(6):pp. 581–595.

Hooker, J. (2002). Logic, optimization, and constraint programming. *INFORMS Journal on Computing*, 14:295–321.

Hooker, J. (2006a). *Handbook of Constraint Programming*, pages 205–239. Elsevier.

Hooker, J. N. (2006b). *Integrated Methods for Optimization (International Series in Operations Research & Management Science)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Hooker, J. N. (2007a). Good and bad futures for constraint programming (and operations research). *Constraint Programming Letters*, 1:21 – 32.

Hooker, J. N. (2007b). Planning and scheduling by logic-based benders decomposition. *Operations Research*, 55(3):588–602.

Hooker, J. N., Ottosson, G., Thorsteinsson, E. S., and Kim, H.-J. (1999). On integrating constraint propagation and linear programming for combinatorial optimization.

IBM (2014). IBM ILOG CPLEX Optimization Studio V12.6. `http://www-01.ibm.com/support/knowledgecenter/SSSA5P/welcome`. [Online; accessed 15-November-2014].

Jain, V. and Grossmann, I. E. (2001). Algorithms for hybrid milp/cp models for a class of optimization problems. *INFORMS Journal on Computing*, 13:258–276.

Kroon, L. G., Salomon, M., and Wassenhove, L. N. V. (1995). Exact and approximation algorithms for the operational fixed interval scheduling problem. *European Journal of Operational Research*, 82(1):190 – 205.

Lombardi, M. and Milano, M. (2012). Optimal methods for resource allocation and scheduling: a cross-disciplinary survey. *Constraints*, 17:51–85.

Lund, J. (1990). Scheduling maintenance dredging on single reach with uncertainty. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 116(2):211–231.

Menon, H. and Lansey, K. (1990). Maintenance scheduling for water resource systems: An application to advance maintenance dredging. Proceedings of the 1990 National Conference, pages 575–579. American Society of Civil Engineers.

Nachtmann, H., Mitchell, K., Rainwater, C., Gedik, R., and Pohl, E. (2014). Optimal dredge fleet scheduling within environmental work windows. *Transportation Research Record: Journal of the Transportation Research Board*, 2426:11 –19.

Pearn, W. L., Chung, S. H., and Yang, M. H. (2002). The wafer probing scheduling problem (wpsp). *The Journal of the Operational Research Society*, 53(8):pp. 864–874.

Pointon, M. (1996). Dredging costs analysis information paper. Technical report, sponsored by the U.S. Army Corps of Engineers Institute for Water Resources, Alexandria, VA,, http://www.wrcndc.itsace.army.mil.

Ratick, S. and Garriga, H. (1996). Risk-based spatial decision support system for maintenance dredging of navigation channels. *Journal of Infrastructure Systems*, 2(1):15–22.

Rodosek, R., Wallace, M., and Hajian, M. (1999). A new approach to integrating mixed integer programming and constraint logic programming. *Annals of Operations Research*, 86:63–87.

Rojanasoonthon, S. and Bard, J. (2005). A grasp for parallel machine scheduling with time windows. *INFORMS Journal on Computing*, 17(1):32–51.

Sadykov, R. and Wolsey, L. A. (Spring 2006). Integer programming and constraint programming in solving a multimachine assignment scheduling problem with deadlines and release dates. *INFORMS Journal on Computing*, 18(2):209–217.

Topaloglu, S. and Ozkarahan, I. (2011). A constraint programming-based solution approach for medical resident scheduling problems. *Computers & Operations Research*, 38(1):246 – 255.

Trick, M. and Yildiz, H. (2007). Benders cuts guided large neighborhood search for the traveling umpire problem. In Van Hentenryck, P. and Wolsey, L., editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 4510 of *Lecture Notes in Computer Science*, pages 332–345. Springer Berlin Heidelberg.

Trick, M. A. and Yildiz, H. (2011). Benders' cuts guided large neighborhood search for the traveling umpire problem. *Naval Research Logistics (NRL)*, 58(8):771–781.

Türsel Eliiyi, D. and Azizoğlu, M. (2009). A fixed job scheduling problem with machine-dependent job weights. *International Journal of Production Research*, 47(9):2231–2256.

Türsel Eliiyi, D. and Azizoğlu, M. (2010). Working time constraints in operational fixed job scheduling. *International Journal of Production Research*, 48(21):6211–6233.

Türsel Eliiyi, D. and Azizoğlu, M. (2011). Heuristics for operational fixed job scheduling problems with working and spread time constraints. *International Journal of Production Economics*, 132(1):107–121.

Van den Akker, J., Hurkens, C., and Savelsbergh, M. (2000). Time-indexed formulations for machine scheduling problems: Column generation. *INFORMS Journal on Computing*, 12(2):111–124.

Van Hoeve, W.-J. and Katriel, I. (2006). *Handbook of Constraint Programming*. Elsevier.