Hybrid Optimization Methods for Time-Dependent Sequencing Problems

Joris Kinable[a,b,*], Andre A. Cire[c], Willem-Jan van Hoeve[a]

[a] *Tepper School of Business, Carnegie Mellon University*
5000 Forbes Ave., Pittsburgh, PA 15213, USA
[b] *Robotics Institute, Carnegie Mellon University*
5000 Forbes Ave., Pittsburgh, PA 15213, USA
[c] *Department of Management, University of Toronto Scarborough*
1265 Military Trail, Toronto, ON M1C 1A4, Canada

Abstract

In this paper we introduce novel optimization methods for sequencing problems in which the setup times between a pair of tasks depend on the relative position of the tasks in the ordering. Our proposed methods rely on a hybrid approach where a constraint programming model is enhanced with two distinct relaxations: One *discrete* relaxation based on multivalued decision diagrams, and one *continuous* relaxation based on linear programming. Both relaxations are used to generate bounds and enhance constraint propagation. Experiments conducted on three variants of the time-dependent traveling salesman problem indicate that our techniques substantially outperform general-purpose methods, such as mixed-integer linear programming and constraint programming models.

Keywords: Constraint Programming, Sequencing, Decision Diagrams, Additive Bounding

1. Introduction

A large number of practical problems in manufacturing, transportation, and distribution require the sequencing of activities over time. The activities may represent jobs to be sequentially processed in an assembly line of a factory, parcel orders of packages to be shipped to customers, or matches in a sports

*Corresponding author; This research was partially carried out when the first author was affiliated with KU Leuven, Department of Computer Science, CODeS & iMinds-ITEC - Belgium.

Email addresses: jkinable@cs.cmu.edu (Joris Kinable), acire@utsc.utoronto.ca (Andre A. Cire), vanhoeve@andrew.cmu.edu (Willem-Jan van Hoeve)

league, to name a few. Sequencing problems have been prominently studied in operational research, in particular in the context of scheduling and routing (Pinedo, 2008; Cook, 2012).

Often activities in a sequencing problem are subject to operational constraints and optimization criteria involving *setup times*, i.e., the minimum time that must elapse between two consecutive activities in a sequence. A setup time typically models the time to change jobs in an assembly line or the travel times between cities in traveling salesman problems (TSPs). In classical sequencing problems (Garey and Johnson, 1979; French, 1982; Pinedo, 2008), the setup time is only defined between pairs of activities. However, in many practical applications the setup time is also a function of the order of the activities in the sequence. Such *position-dependent* setup times are useful in modeling different states of a resource throughout a schedule, such as when aging and learning effects are considered (Rudek, 2012). Aging effect takes place when the internal components of a machine degrades after performing a number of tasks, or due to fatigue of human resources throughout a day; see, e.g., Jeng and Lin (2004); Yin et al. (2012); Huang and Wang (2015). Learning effect occurs when an employee becomes more proficient in her tasks, or in highly automated manufacturing system when iterative control systems compensate for motion errors after each job is completed (Arimoto et al., 1984; Biskup, 1999; Wu et al., 2007; Yin et al., 2009).

Another well-known example of application is the *traveling deliveryman problem* (TDP), associated with several practical applications in operations research (Blum et al., 1994; Simchi-Levi and Berman, 1991). Given a set of geographically dispersed customers and a central depot, the TDP asks for a tour that starts at the depot and minimizes the total sum of times to arrive at each customer. The TDP can be cast as a position-dependent scheduling problem through a suitable transformation (Lucena, 1990), and spans a variety of exact approaches in the literature; see, e.g., Fischetti et al. (1993); van Eijl (1995); Méndez-Díaz et al. (2008).

Introducing position-dependent setup times typically makes the problem much more difficult to solve when compared to the position-independent counterpart. While Applegate et al. (2006a) reports solving the classical TSP with up to 100,000 nodes, only recent attempts were successful in solving 100-node instances for the TSP variant with position-dependent travel times (Abeledo et al., 2013). In addition, literature that considers typical real-world side constraints, such as precedence relations or time windows, is noticeably limited. In

this case, the majority of recent publications focus on dedicated solution techniques or heuristics. We refer to the recent survey by Gendreau et al. (2015) for more details.

In this paper we investigate efficient generic approaches to solve position-dependent sequencing problems which can easily incorporate additional side constraints, such as precedence constraints and time windows. Our techniques are based on exact hybrid methods that combine linear programming (LP), constraint programming (CP), and *multivalued decision diagrams* (MDDs), each having complementary strengths that can be exploited in novel solution methods for this class of problems.

Specifically, our proposed models are primarily solved within a CP framework, which is well-suited for a variety of complex scheduling problems due to its language expressiveness and specialized inference and search mechanisms (Baptiste et al., 2006). However, classical CP approaches are typically ineffective when solving sequencing problems with setup times, in particular due to the fact that the problem constraints are entirely decoupled from the objective function, leading to ineffective bounds (Andersen et al., 2007). To counteract these issues, we propose to enhance CP solution techniques with both a *discrete* and a *continuous relaxation*, described as follows.

The discrete relaxation is obtained by using MDDs, a graphical structure derived from a dynamic programming formulation of the problem and which have recently been proven highly effective for scheduling (Cire and van Hoeve, 2013). MDDs provide a compact encoding of the solution space through which both lower and upper bounds on the optimum solution value can be computed. Specifically, MDDs enable computing bounds at each node of the CP search tree, thereby providing a means to effectively prune and guide the search.

The continuous relaxation of the problem is motivated by the extensive literature on mixed-integer linear programming (MILP) formulations for sequencing problems (Queyranne et al., 1994; van den Akker et al., 1999; Keha et al., 2009). Each formulation yields a continuous relaxation, i.e., the LP relaxation obtained by relaxing the integrality constraints. The LP provides bounds as well as dual information that can be exploited in a number of ways in optimization techniques. Fischetti and Toth (1989) proposed an *additive bounding procedure* which combines different LP relaxations to obtain a new valid optimization bound for the problem at hand, generally stronger than the strongest bound obtainable from the individual relaxations. In this work we demonstrate how additive bounding can be used to incorporate dual information from LP

relaxations into the MDDs, which can substantially strengthen the MDD.

In order to demonstrate the effectiveness of our hybrid method and to simplify exposition, we focus on three variants of the *time-dependent traveling salesman problem* (TD-TSP), the generalization of the TSP in which the travel time between two cities depend on the order in which cities are visited. Several solution approaches for the TD-TSP problem have been proposed, including a number of techniques based on MILP (Picard and Queyranne, 1978; Gouveia and Voss, 1995; Miranda-Bront et al., 2010; Abeledo et al., 2013). Moreover, to exemplify the generality of our method, we also study the *TD-TSP with time windows* (TD-TSPTW) and the *time-dependent sequential ordering problem* (TD-SOP). The TD-SOP is identical to the TD-TSP, except for the fact that this problem also embodies precedence relations between tasks. We show that our proposed approaches are consistently superior to other generic techniques for both problems.

We note in passing that other related problems are also denoted by *time-dependent TSP* in the literature. For instance, Cordeau et al. (2012) investigate TSP problems with time-dependent travel speeds, while travel times in Albiach et al. (2008) and Malandraki and Daskin (1992) are defined as a function of the instant the tour leaves each city. In this paper we focus on the position-dependent version.

Contributions. Our two main contributions are as follows.

1. As our major result, we advance the state of the art in general-purpose position-dependent scheduling by proposing new solution methods to tackle such a class of problems. One of the key benefits of our method is that it can naturally incorporate a number of practical side constraints, such as time windows and precedence constraints.
2. We propose a novel way on how discrete and continuous relaxations can be integrated to speed up solution techniques, in particular using MDDs and LP relaxations. This technique also reveals a number of opportunities to integrate other well-studied relaxations for sequencing problems, such as the Held-Karp relaxation (Held and Karp, 1970), assignment relaxations, and Lagrangian relaxations.

A number of state-of-the-art solution approaches for each of the individual problems studied in this paper have appeared previously in the literature. Abeledo et al. (2013) developed a highly efficient column generation approach

for TD-TSP, strengthened with valid inequalities. Similarly, Baldacci et al. (2012) proposed a sophisticated method to solve the TSP-TW, a special case of the TDTSP-TW, through a dynamic programming approach with various state space relaxations. Gouveia and Ruthmair (2015) and Cire and van Hoeve (2013) recently closed a number of long-standing SOP instances. Gouveia and Ruthmair (2015) relies on a cutting plane algorithm embedded in a branch-and-bound framework, whereas Cire and van Hoeve (2013) use MDDs. Because exact methods for time-dependent versions of TSP-TW and SOP have not been studied before, we compare our approach with other generic solution methods, based on MIP and CP technology. As we will see in Section 6, our approach substantially outperforms CP and MIP models, often by orders of magnitude improved solving time, although on pure TD-TSP instances the specific approach of Abeledo et al. (2013) generally outperforms our method.

In the remainder of this paper, Section 2 gives a formal description of the TD-TSP and of the TD-SOP. Next, CP models for both problems are provided in Section 3. Section 4 introduces MDDs and discusses how they are integrated in a CP model for the TD-TSP. Then, Section 5 focuses on additive bounding, and shows how to consolidate the bounds into an MDD. Computational experiments conducted on the TD-TSP and TD-SOP are reported in Section 6, where generic MILP and CP approaches are compared to our hybrid approach. Finally, Section 7 concludes the paper.

2. The TD-TSP(TW) and TD-SOP Problem

The TD-TSP is a version of the TSP where travel times also depend on the order in which each city is being visited. In this paper we treat the TD-TSP as a scheduling problem. Let $J = \{1\} \cup \{2, \dots, n-1\} \cup \{n\}$ be a set of n jobs that need to be sequenced on a single non-preemptive machine, where jobs 1 and n must be the first and last job in the sequence, respectively. With each pair of jobs $i, j \in J$ and parameter $t \in \{1, \dots, n\}$ we associate a *setup time* $\delta^t_{i,j} \in \mathbb{R}$. The goal of the TD-TSP is to find a sequence of jobs $(\pi_1, \dots, \pi_n)$ where $\pi_1 = 1$, $\pi_n = n$, $\pi_t \in J$ for all $t = 1, \dots, n$, $\pi_t \neq \pi_q$ for all $t, q = 1, \dots, n$, $t \neq q$, and such that the total sum of setup times,

$$\sum_{t=1}^{n-1} \delta^t_{\pi_t, \pi_{t+1}},$$

is minimized. We do not assume that δ^t is symmetric, i.e., we may have $\delta^t_{i,j} \neq \delta^t_{j,i}$ for some t and pair of jobs $i, j \in J$.

The TD-TSPTW is an extension of the TD-TSP in which with each job j a time window $[a_j, b_j]$ is associated during which the job must be executed.

Finally, the TD-SOP is a generalization of the TD-TSP in which a set P of precedence relations between pairs of jobs must be observed. In particular, the relation $i \prec j \in P$ denotes that job i must precede job j in any sequence, i.e., if $\pi_t = i$ and $\pi_q = j$ in a feasible sequence $(\pi_1, \ldots, \pi_n)$, then $t < q$.

3. Constraint Programming Models

The basis of our approach consists of formulating the problem as a CP model. Constraint programming offers a flexible modeling language in which problem structure is captured through special constructs denoted by *global constraints*. A global constraint may indicate to solvers how to perform inference, decompose the problem, or guide the search based on the substructure they represent (Régin, 2011).

To formulate the TD-TSP as a CP model, we introduce variable π_t representing the job performed at position t in the sequence, for $t = 1, \ldots, n$. The CP model is stated as follows.

$$
\begin{aligned}
\text{minimize} \quad & \sum_{t=1}^{n-1} \delta^t_{\pi_t, \pi_{t+1}} \\
\text{s.t.} \quad & \texttt{alldifferent}(\pi_1, \ldots, \pi_n), && \text{(CP)} \\
& \pi_1 = 1, \quad \pi_n = n, \\
& \pi_t \in \{2, \ldots, n-1\}, && t = 2, \ldots, n-1.
\end{aligned}
$$

The `alldifferent` global constraint imposes that variables $\pi_1, \ldots, \pi_n$ are pairwise distinct and ensures that each job is performed exactly once. Note that in the objective function the term $\delta^t_{\pi_t, \pi_{t+1}}$ uses variables as subscripts. These are efficiently handled by *element* constraints in CP (Hooker, 2012).

We can augment this model to the TD-TSPTW by introducing variables u_i representing the completion time of the job performed at position i of the sequence. The following channeling constraints are added to link the u_i and the

π_t variables, and to enforce the time windows $[a_j, b_j]$ for each job j:

$$\begin{aligned} & u_1 = a_1 \\ & u_t = \max\{a_{\pi_t}, u_{t-1} + \delta^{t-1}_{\pi_{t-1}\pi_t}\} && t = 2, \ldots, n-1, \\ & a_{\pi_t} \leq u_t \leq b_{\pi_t} && \forall t = 1, \ldots, n \end{aligned} \tag{TD-TSPTW}$$

Finally, we can extend the TD-TSP model to the TD-SOP by introducing variables l_i represent the position of job i in the sequence, for $i = 1, \ldots, n$. The following channeling constraints are added to link the l_i and π_t variables, and to impose the precedence constraints:

$$\begin{aligned} & l_{\pi_t} = t, && t = 1, \ldots, n, \\ & l_i \leq l_j - 1, && \forall\, i \prec j \in P, \\ & l_i \in \{1, \ldots, n\} && i = 1, \ldots, n. \end{aligned} \tag{CP-SOP}$$

A CP model is typically solved by a backtracking search coupled with individual constraint processing. Namely, at every CP search node the constraints are processed one at a time, each eliminating values from the variable domains that lead to infeasible or suboptimal solutions. This is denoted by *filtering*. Once constraint processing is done, CP solution techniques branch on variables by partitioning their domains. We refer to the book by Hooker (2012) for more details on the CP search and inference mechanism.

3.1. Practical Difficulties

Despite the fact that the formulations above offer a valid formulation for the TD-TSP(TW) and TD-SOP, traditional CP solution techniques are ineffective for reasonably-sized problem instances. This can be attributed to the following three causes:

1. Each constraint in CP is comparable to a black box, implementing its own filtering mechanism. Communication between constraints is solely achieved through the variable domains, which can lead to a significant loss of structural information. Examples of this are presented by Andersen et al. (2007).
2. The `alldifferent` constraint only ensures that each job is scheduled exactly once, but does not take the objective function into consideration. This limits the effectiveness of the domain filtering in identifying suboptimal solutions.

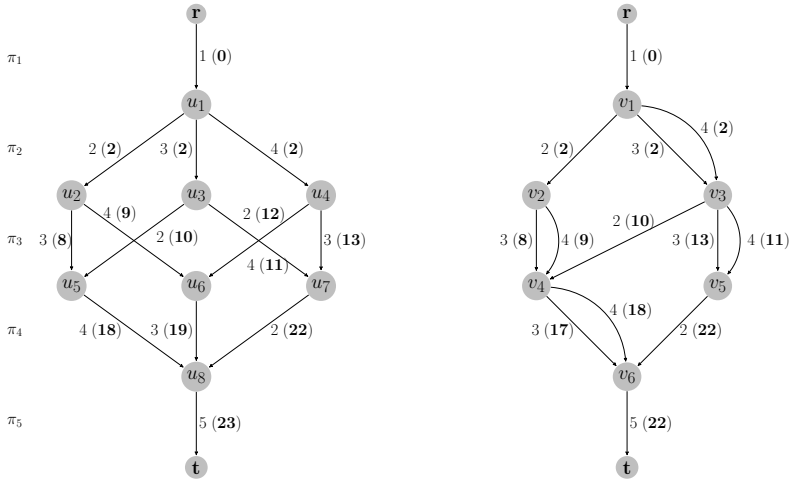
3. CP solvers typically do not employ any type of problem relaxation and, thus, have little to no means to derive strong bounds on the optimum solution, prohibiting effective pruning of the CP search tree for optimization problems.

The following sections describe two types of relaxations that can be incorporated into a CP solution technique to address the causes above. Each relaxation provides a distinct and complementary global perspective of the problem, yielding novel optimization bounds and filtering mechanisms.

4. Discrete Relaxation based on MDDs

An MDD for a sequencing problem is a graphical structure that encodes a set of job sequences for a problem instance (Cire and van Hoeve, 2013). Specifically, an MDD is a directed acyclic layered graph in which the nodes are partitioned into $n+1$ layers $L_1, \ldots, L_{n+1}$. Layers L_1 and L_{n+1} are singletons containing the root node $\mathbf{r}$ and terminal node $\mathbf{t}$, respectively. There is a one-to-one correspondence between each layer L_t, $t = 1, \ldots, n$, and the t-th job to be performed on the machine (i.e., layer L_t corresponds with variable π_t in the CP model of Section 3). With each arc a in the graph we associate a label $val(a) \in J$. An arc a with label $val(a)$, leaving a node in layer L_t, corresponds to assigning job $val(a)$ to the t-th position of the machine (i.e., $\pi_t = val(a)$). Thus, an arc-specified path $(a_1, \ldots, a_n)$ from the nodes $\mathbf{r}$ to $\mathbf{t}$ defines the job sequence $(\pi_1, \pi_2, \ldots, \pi_n) = (val(a_1), val(a_2), \ldots, val(a_n))$. Examples of MDDs are given in Figure 1 for a set of jobs $J = \{1, 2, 3, 4, 5\}$ (ignore for now the numbers in parenthesis). Note, e.g., that the path through nodes $(\mathbf{r}, u_1, u_4, u_7, u_8, \mathbf{t})$ yields the sequence $(1, 4, 3, 2, 5)$. The number of nodes in a layer, $|L_t|$, is denoted by the *width* of the layer. The width of the MDD is the maximum width among all layers. For example, in Figure 1a the MDD width is 3.

An MDD is *exact* if each path from $\mathbf{r}$ to $\mathbf{t}$ corresponds to a feasible sequence, and all feasible sequences are encoded by some path in the graph. Figure 1a depicts an exact MDD for a TD-TSP instance with 5 jobs. An MDD is *relaxed* if all feasible sequences are represented by some path in the graph, but not all paths in the graph necessarily encode feasible sequences. In particular, a relaxed MDD for the TD-TSP may contain sequences in which a job is performed more than once, and for the TD-SOP it may also contain sequences that violate the precedence constraints. For example, Figure 1b depicts a relaxed 2-width MDD



(a) 3-width MDD (exact). (b) 2-width MDD (relaxed).

Figure 1: Examples of MDD for a sequencing problem with jobs $J = \{1, 2, 3, 4, 5\}$. The label of an arc a is read as $val(a)$ $(s(a))$, where $val(a) \in J$ is the job the arc assigns and $s(a)$, the number in parenthesis, is computed according to relation (1).

which contains a path through nodes $(\mathbf{r}, v_1, v_2, v_4, v_6, \mathbf{t})$ encoding the infeasible sequence $(1, 2, 3, 3, 5)$.

Exact and relaxed MDDs can be used to compute optimization bounds for typical scheduling objective functions (Cire and van Hoeve, 2013). Such bounds are obtained by evaluating recursive relations over the graphical structure of the MDD. In particular, given an exact or relaxed MDD for a TD-TSP instance, a lower bound on the sum of setup times is computed as follows. Let $in(u)$ be the set of incoming arcs at a node u and $\ell(a)$ the layer index of the source node of arc a; e.g., in Figure 1a we have $in(u_5) = \{(u_2, u_5), (u_3, u_5)\}$ and $\ell((u_5, u_8)) = 4$. Recall that $\delta^t_{i,j}$ represents the setup time between jobs i and j when job i occurs at position t in the sequence. We define the *setup of an arc* a, $s(a)$, as the minimum sum of setup times of all subsequences identified by paths ending at a. For an arc $a = (u, v)$, this can be written recursively as

$$s(a) = \begin{cases} 0 & \text{if } u = \mathbf{r}, \\ \min\limits_{a' \in in(u), val(a) \neq val(a')} \{s(a') + \delta^{\ell(a')}_{val(a'), val(a)}\} & \text{otherwise.} \end{cases} \tag{1}$$

The validity of the recursive relation (1) follows from Bellman's principle of optimality, since the setup time to start a job i depends only on the previous job and its position in the sequence. It follows that a lower bound on the sum

of setup times is given by

$$\min_{a \in in(\mathbf{t})} s(a), \tag{2}$$

which corresponds to the optimal solution value if the MDD is exact. The sequence that evaluates to the objective value given by (2) can be recovered by traversing the diagram from node $\mathbf{t}$ to $\mathbf{r}$, following back the arcs that were selected in the minimization term of relation (1).

As an illustration, consider the instance with jobs $J = \{1, \ldots, 5\}$ depicted by the MDDs in Figure 1. Let the setup times be defined by $\delta^t_{i,j} = \delta_{(i,j)} + t$, where $\delta_{i,j}$ is given by the following table.

	1	2	3	4	5
1	-	1	1	1	-
2	-	-	4	5	1
3	-	6	-	7	1
4	-	8	9	-	1
5	-	-	-	-	-

The number in parenthesis next to an arc in Figures 1a and 1b represents the value of $s(a)$ according to relation (1). The optimal solution in Figure 1a is given by the sequence of jobs $(1, 2, 3, 4, 5)$ and has value 23, obtained by the path that traverses nodes $(\mathbf{r}, u_1, u_2, u_5, u_8, \mathbf{t})$. In the relaxed MDD of Figure 1b, the optimal sequence is given by $(1, 3, 2, 3, 5)$ and provides a lower bound of 22, obtained the path that traverses nodes $(\mathbf{r}, v_1, v_3, v_4, v_6, \mathbf{t})$.

4.1. Constructing Relaxed MDDs

For practical instances, an exact MDD is often too large to process efficiently or to fit into memory. Instead we use relaxed MDDs of limited width W to efficiently calculate bounds on the optimal solution value and to perform inference, e.g. to deduce precedence relations between jobs. The quality of the relaxation is controlled by parameter W: larger values allow us to obtain stronger bounds, albeit at a higher computational cost.

A relaxed MDD for a sequencing problem is compiled through an iterative procedure denoted by *incremental refinement* (Andersen et al., 2007; Cire and van Hoeve, 2013). Given a maximum width $W \geq 1$, the procedure starts with a trivial 1-width relaxed MDD that is valid for any sequencing problem, depicted in Figure 2a. Except for the first and last job in the sequence, the 1-width MDD encodes all possible sequences of jobs with repetitions. The compilation procedure then performs two operations at each iteration, *refinement* and *filtering*:

- *Refinement* makes the diagram representation coarser by heuristically selecting and *splitting* a node. The split of a node consists of partitioning the incoming arcs of a node into two sets, redirecting the arcs of one of the sets into a new node added to the same layer, and replicating the outgoing arcs of the split node into the new node.

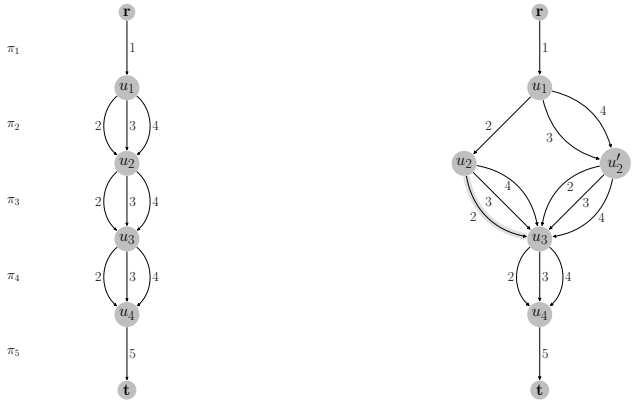
  For example, Figure 2b depicts the split of node u_2 from the MDD in Figure 2a. The incoming arcs of node u_2 are partitioned into two sets, one having label $\{2\}$ and the other with $\{3, 4\}$. The arcs with labels $\{3, 4\}$ are redirect to u'_2, and the outgoing arcs from u_2 are replicated in u'_2. Note that the set of solutions does not change after a node splitting.

- *Filtering* strengthens the relaxation represented by the MDD by removing *infeasible arcs*. An arc is infeasible if all paths traversing the arcs identify infeasible sequences (or suboptimal solutions, if an upper bound to the problem is known). Removing an arc can potentially eliminate an exponential number of infeasible sequences from the MDD.

  For example, in Figure 2b the shaded arc from node u_2 to u_3 is infeasible, since the paths traversing that arc identify sequences for which job 2 is performed at least twice. The same arc is not infeasible in Figure 2a.

Refinement and filtering are performed iteratively until no more nodes can be split (otherwise the maximum width W would be violated) and filtering is unable to remove any arcs. In general sequencing problems, identifying sufficient conditions for the infeasibility of arcs is NP-Hard, hence we restrict our attention to necessary infeasibility tests. Moreover, node splitting operation is mainly heuristic, as often there are more candidate nodes to refine than the width of the MDD permits.

In this work, we use the same refinement and filtering operations proposed by Andersen et al. (2007) and specialized for sequencing problems in Cire and van Hoeve (2013). The intuition is to perceive the diagram as a state-transition graph for a valid dynamic programming formulation of the problem, and eliminate arcs that correspond to infeasible state transitions. Moreover, we can also split nodes that implicitly represent two or more states merged together. Formally, this is done as follows. With each node u of the MDD we associate two states, $All(u)$ and $Some(u)$, representing the jobs that appear in *all* paths from $\mathbf{r}$ to u and those that appear in *at least one* path from $\mathbf{r}$ to u, respectively. For example, in Figure 2b we have $Some(u'_2) = \{1, 3, 4\}$ and $All(u'_2) = \{1\}$. Intu-



(a) 1-width MDD.

(b) After refinement.

Figure 2: Example of refinement and filtering. The shaded arc in Figure 2b represents an infeasible arc identified by filtering, and can be removed from the graph.

itively, each state indicates jobs that were necessarily performed or that may have been performed in the subsequences so far.

A number of filtering tests based on the states above are described in the work by Andersen et al. (2007). For instance, in Figure 2b the arc with label 2 from u_2 to u_3 can be removed because $2 \in All(u_2)$ (i.e., the job has already been performed in all subsequences up to u_2). For refinement, Cire and van Hoeve (2013) show that, in any exact MDD for a sequencing problem, we have $Some(u) = All(u)$ for all nodes u. A refinement heuristic then tries to satisfy this condition by splitting the first node where this condition is violated.

We refer to the work of Cire and van Hoeve (2013) for a complete description of the filtering tests and the refinement heuristics, as well as filtering tests for precedence constraints and time windows.

4.2. CP model with MDD

To incorporate an MDD relaxation into a CP solver, we add a new global constraint to the models described in Section 3, namely

$$\texttt{MDDConstraint}(\pi_1, \ldots, \pi_n, W, \delta, P, T).$$

The constraint takes as input the variables $\pi_1, \ldots, \pi_n$, the maximum allowed width W of the MDD, the position-dependent setup times δ, the set of precedence relations P, the time windows T. We note that, although the constraint

above suffices to enforce that the variables π are pairwise distinct, it is computationally advantageous for the CP solver to have the `alldifferent` constraint redundantly in the model. This is due to the fact that classical CP filtering techniques for the `alldifferent` enforce arc consistency (Régin, 1994) in low-order polynomial time, and thus are capable of efficiently identifying and removing all infeasible variable-value assignments with respect to this constraint.

The `MDDConstraint` is processed at every search node of the CP search tree. First, the domains of the variables $\pi_1, \dots, \pi_n$ are synchronized with the MDD, thereby possibly removing arcs from the MDD. For example, if a job $j \in J$ is fixed to position t in the sequence, i.e., $\pi_t = j$, then all arcs a with label $val(a) \neq j$ leaving nodes in layer L_t are removed. The MDD then goes through the process of filtering and refinement described in Section 4.1. A lower bound is then computed and provided to the solver, which will be used to prune suboptimal branches.

In addition, if an upper bound z^* is provided by the solver, then we remove all arcs in the MDD which are necessarily traversed by suboptimal paths. According to relation (1), these are the arcs a such that $s(a) > z^*$.

5. Continuous Relaxation and Additive Bounding

For most optimization problems, several relaxations exist based on different underlying (combinatorial) problem structures. For instance, common relaxations for the TSP are the Held and Karp (1970) bound, the assignment problem relaxation, and strengthened LP relaxations (Applegate et al., 2006b). To obtain a valid bound on the optimum solution, one could simply compute a bound from each relaxation, and return the strongest resulting bound. The disadvantage of such an approach is that the structural information of only one relaxation is used. To resolve this issue Fischetti and Toth (1989) proposed an *additive bounding* procedure which aggregates the information from different relaxations to obtain a bound at least as strong as the tightest relaxation. We show how this procedure can be used to incorporate information from an LP relaxation into the structure of an MDD, though other relaxations may be used as well. The resulting approach provides a number of merits:

- Integrating information from different problem relaxations into the structure of the MDD enables additional filtering conditions.

- The MDD provides an *interface* to incorporate structural information from various problem relaxations into the CP model.
- MDDs provide a discrete relaxation of the solution space, while LPs produce a continuous relaxation. Projecting structural information from a continuous relaxation onto a discrete structure potentially improves the overall strength of the relaxation.

5.1. Additive bounding

The additive bounding procedure proposed by Fischetti and Toth (1989) offers a means to compute valid optimization bounds for a problem by combining multiple bounding procedures. Namely, let $\mathcal{P} := \{\min\ cx\ :\ x \in X\}$ for some cost vector c and $X \subseteq \mathbb{R}^n_*$. Suppose that, for an arbitrary cost vector $\bar{c}$, r distinct lower bound procedures $L^1(\bar{c}), L^2(\bar{c}), \ldots, L^r(\bar{c})$ are available. Each $L^k(\bar{c})$ produces a lower bound μ^k on the optimal value of $\mathcal{P}$ as well as a *residual cost* $c^k \geq 0$ satisfying $\mu^k + c^k x \leq \bar{c}x$ for all $x \in X$.

The additive bounding procedure starts by calculating bound μ^1 and residual cost vector c^1 through bounding procedure $L^1(\bar{c})$, where $\bar{c}$ is the original cost vector in the problem instance. Next, $L^k(c^{k-1})$ for $k = 2, \ldots, r$ can be solved recursively, thereby obtaining bounds $\mu^2, \ldots, \mu^r$. Fischetti and Toth (1989) prove that $\mu = \mu^1 + \cdots + \mu^r$ is a valid lower bound for problem P, which is potentially stronger than each individual bound.

The use of additive bounding in the context of Constraint Programming is not new. Lodi et al. (2006) use additive bounding to accelerate the proof of optimality in a Discrepancy Based Search framework. Furthermore, Benchimol et al. (2012) present a global weighted circuit constraint; the filtering mechanism for the weighted circuit constraint relies on a number of problem relaxations, which are combined through the aforementioned additive bounding procedure.

5.2. Strengthening MDD Relaxations

After solving the additive bounding procedure with k lower bounding procedures, one obtains a bound μ^k and a final residual cost vector c^k. Assume that we have such a lower bound μ for TD-TSP and TD-SOP, and corresponding residual costs $\Delta^t_{i,j}$ for jobs $i, j \in J$, and positions $t = 1, \ldots, n$. Recall that for the residual cost vector, the following relation must hold: $\mu^k + c^k x \leq \bar{c}x$. In case of TD-TSP/SOP, and for some path $(\pi_1, \ldots, \pi_n)$, this becomes:

$$\mu + \sum_{t=1}^{n-1} \Delta^t_{\pi_t, \pi_{t+1}} \leq \sum_{t=1}^{n-1} \delta^t_{\pi_t, \pi_{t+1}} \tag{3}$$

Similar to relation 1 in Section 4, we can calculate a lower bound on the sum of residual costs incurred on a path ending with an arc $a = (u, v)$ as

$$\overline{s}(a) = \begin{cases} 0 & \text{if } u = \mathbf{r}, \\ \min\limits_{\substack{a' \in in(u), \\ val(a) \neq val(a')}} \{\overline{s}(a') + \Delta^{\ell(a')}_{val(a'),val(a)}\} & \text{otherwise.} \end{cases} \tag{4}$$

A bound on the sum of residual costs on a path from $\mathbf{r}$ to $\mathbf{t}$ can be obtained by evaluating $\min_{a \in in(\mathbf{t})} \overline{s}(a)$. Substituting this into equation (3) yields the following valid condition: $\mu + \min_{a \in in(\mathbf{t})} \overline{s}(a) \leq z^*$, where z^* is the objective value of any feasible solution to our problem. In other words, $\mu + \min_{a \in in(\mathbf{t})} \overline{s}(a)$ is a valid lower bound on our problem. Additionally, if, in the relaxed MDD, there exists an arc a such that *any* path from $\mathbf{r}$ to $\mathbf{t}$ *through* arc a violates the aforementioned condition, then this arc may be filtered from the MDD. To implement this filtering rule, we use the same technique used to filter objective function bounds presented by Cire and van Hoeve (2013). In this work, z^* is directly obtained from the CP solver; the residual costs are the reduced costs corresponding with the variables in the Linear Programming relaxation presented in the next subsection. Note that only the reduced costs of the variables which are not at their upper bound are used; for the remaining variables the reduced cost is set equal to zero.

Finally, note that the additive bounding procedure can be executed as a preprocessing step at the root node of the CP search tree, since the bounds are independent of the MDD or the CP model. Consequently, the computational overhead is very limited if the lower bound procedure is efficient.

5.3. LP Relaxation

We now present the LP relaxations for the three applications investigated in this paper.

5.3.1. TDTSP

In this work we consider a single bounding procedure for the TD-TSP and TD-SOP: the LP relaxation of the MILP model by Vander Wiel and Sahinidis (1995) which is a linearization of the quadratic assignment problem presented by Picard and Queyranne (1978). The formulation is based on a time-space network as depicted in Figure 3, where each node (i, t) represents a job i performed at position t. The notation is made consistent with Miranda-Bront et al. (2014).

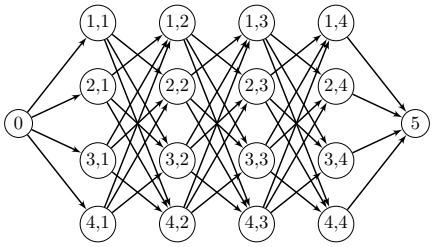


Figure 3: Time Space Network for the TD-TSP on five cities Picard and Queyranne (1978). State (i, t) represents visiting city i in position t.

$$\text{MILP: } \min \sum_{j=1}^{n-1} \delta_{0,j}^{0} y_{0,j}^{0} + \sum_{i=1}^{n-1} \sum_{\substack{j=1 \\ j \neq i}}^{n-1} \sum_{t=1}^{n-2} \delta_{ij}^{t} y_{ij}^{t} + \sum_{i=1}^{n-1} \delta_{i,n}^{n-1} y_{i,n}^{n-1} \tag{5}$$

$$\text{s.t. } \sum_{t=1}^{n-1} x_{it} = 1 \qquad \forall i = 1, \ldots, n-1 \tag{6}$$

$$\sum_{i=1}^{n-1} x_{it} = 1 \qquad \forall t = 1, \ldots, n-1 \tag{7}$$

$$y_{0j}^{0} = x_{j1} \qquad \forall j = 1, \ldots, n-1 \tag{8}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n-1} y_{ij}^{t} = x_{it} \qquad \forall j = 1, \ldots, n-1, t = 1, \ldots, n-2 \tag{9}$$

$$y_{in}^{n-1} = x_{i,n-1} \qquad \forall i = 1, \ldots, n-1 \tag{10}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n-1} y_{ij}^{t-1} = x_{jt} \qquad \forall j = 1, \ldots, n-1, t = 2, \ldots, n-1 \tag{11}$$

$$\sum_{j=1}^{n-1} y_{0j}^{0} = 1 \tag{12}$$

$$\sum_{j=1}^{n-1} x_{jn}^{n-1} = 1 \tag{13}$$

$$x_{it} \in \{0, 1\} \tag{14}$$

$$y_{ij}^{t} \geq 0 \tag{15}$$

In this model *transition* variables y_{ij}^{t} represent whether job $j \in J$ is performed immediately after job $i \in J$, which is performed at time t. *Assignment* variables x_{it} denote whether job $i \in J$ is performed at time t. Vander Wiel and Sahinidis (1995) prove that $y_{ij}^{t} = 1$ if and only if $x_{i,t} x_{j,t+1} = 1$.

Constraints (6) and (7) ensure resp. that each job is performed only once, and that at any given point in time only one job is performed. Constraints (8)-

(11) link the x_{it} and y^t_{ij} variables. Moreover, Constraints (8)-(11) implement flow conservation on the time-space network. Finally, Constraints (12), (13) ensure that exactly one unit of flow resp. leaves the source node and enters the sink node. Consistent with the models from the previous sections, we enforce that job 0 starts at time 0, and job n starts at time n. Consequently variables x_{00} and x_{nn} are fixed to 1.

Several possible approaches to strengthen this model are presented by Gouveia and Voss (1995) and Abeledo et al. (2013). Most notably, the time-dependent TSP can be mapped to the traditional time-independent TSP through the following relation:

$$z_{ij} = \sum_{t=1}^{n} (y^t_{ij} + y^t_{ji}) \qquad \forall i, j = 0, \ldots, n \tag{16}$$

Consequently, all valid inequalities known for the TSP can also be used to strengthen this TD-TSP model. Furthermore, for any cyclic sequence of unique jobs $C = [v_1, v_2, \ldots, v_l, v_{l+1} = v_1]$, $v_i \in J$, Miranda-Bront et al. (2014) show that the following *time-dependent cycle inequalities*, also known as k-cycle inequalities for $k = |C|$, are valid for the MILP model of the TD-TSP:

$$y^t_{v_1 v_2} + y^{t+1}_{v_2 v_3} + \cdots + y^{t+k-2}_{v_l v_1} \leq x^t_{v_1} + x^{t+1}_{v_2} + \cdots + x^{t+k-2}_{v_l} \quad 0 < t < n - k \tag{17}$$

For the residual costs in the additive bounding procedure, we directly use the reduced costs of the y^t_{ij} ($\overline{y}^t_{ij}$) and x_{it} ($\overline{x}_{it}$) variables which are not at their upper bound, by setting $\Delta^t_{ij} = \overline{y}^t_{ij} + \overline{x}_{it}$ in Equation (4).

We note in passing that other formulations for the TDTSP have been described in the literature, such as the one proposed by Godinho et al. (2014). We consider the MILP model above due to the strengthening procedure by Miranda-Bront et al. (2014) and since they can be more easily adapted to consider additional side constraints.

5.3.2. TDTSP-TW

In the TDTSP-TW, a time-window $[a_i, b_i]$ is associated with each job $i \in J$, during which the job has to be performed. The MILP model presented in Section 5.3.1 can be adapted to incorporate time windows, thereby obtaining a model for TDTSP-TW. Let the variable u_i denote the completion time of a job $i \in J$.

Then the following constraints enforce the time-windows:

$$u_0 + \delta^0_{0j} y^j_{0j} \leq u_j \qquad \forall i = 1, \ldots, n-1 \tag{18}$$

$$u_i + \delta^{n-1}_{i,n} y^{n-1}_{i,n} \leq u_n \qquad \forall i = 1, \ldots, n-1 \tag{19}$$

$$u_i + \sum_{t=1}^{n-2} \delta^t_{i,j} y^t_{i,j} - M_{ij}(1 - \sum_{t=1}^{n-2} y^t_{i,j}) \leq u_j \qquad \forall i, j = 1, \ldots, n-1, i \neq j \tag{20}$$

$$a_i \leq u_i \leq b_i, \tag{21}$$

where $M_{ij} = \max_t\{0, b_i + \delta^t_{ij} - a_j\}$.
Constraints (20)-(21) are strengthened as follows (Desrochers and Laporte, 1991):

$$u_j \geq a_j + \sum_{t=0}^{n-1} \sum_{\substack{i=0 \\ i \neq j}}^{n-1} \max\{0, a_i + \delta^t_{ij} - a_j\} y^t_{ij} \qquad \forall j = 1, \ldots, n \tag{22}$$

$$u_i \leq b_i - \sum_{t=0}^{n-1} \sum_{\substack{j=1 \\ i \neq j}}^{n} \max\{0, b_i + \delta^t_{ij} - b_j\} y^t_{ij} \qquad \forall i = 0, \ldots, n-1 \tag{23}$$

$$u_j \geq u_i + \sum_{t=1}^{n-2} \delta^t_{i,j} y^t_{i,j} - M_{ij}(1 - \sum_{t=1}^{n-2} y^t_{i,j}) + \qquad \forall i, j = 1, \ldots, n-1, i \neq j \tag{24}$$

$$\sum_{t=1}^{n-2} (M_{ij} - \delta^t_{i,j} - \max\{\delta^t_{ji}, b_i - b_j\}) y^t_{ji} \tag{25}$$

For the residual costs in the additive bounding procedure, we use $\Delta^t_{ij} = \overline{y}^t_{ij} + \overline{x}_{it}$ (Equation (4)), where $\overline{y}^t_{ij}$, $\overline{x}_{it}$ are respectively the reduced costs corresponding with the y^t_{ij}, x_{it} variables which are not at their upper bound.

We note in passing that the TDTSP-TW shares some similarities with the *delivery man problem with time windows* (DMP-TW), as studied by Heilporn et al. (2010). The objective of the DMP-TW is to minimize the sum of the travel durations to each city, i.e., the differences between the times each city is visited and when the salesperson left the depot. While the classical delivery man problem without time windows can be written as an instance of the TDTSP (Blum et al., 1994), the same is not true between the DMP-TW and the TDTSP-TW, in particular since the release date of the depot also becomes a variable in the case of the DMP-TW.

5.3.3. TDSOP

To accommodate precedence relations for the TD-SOP, we add constraints from a MILP model by Sarin et al. (2005) for the asymmetric TSP (ATSP) with

precedence constraints. The model by Sarin et al. (2005) is known to provide a strong LP relaxation. To accommodate the precedence constraints, p_{ij} variables are used, indicating whether job i precedes job j *somewhere*, but not necessarily immediately, in the sequence.

$$p_{ij} \geq \sum_{t=0}^{n-1} y_{ij}^t \qquad \forall i, j = 1, \ldots, n-1, i \neq j \tag{26}$$

$$p_{ij} + p_{ji} = 1 \qquad \forall i \neq j \tag{27}$$

$$p_{ij} + p_{jk} + p_{ki} \leq 2 \qquad \forall i \neq j \neq k \tag{28}$$

$$y_{ik}^t = 0 \qquad \text{iff } i \prec j \prec k, \forall t = 1, \ldots, n-1 \tag{29}$$

$$x_{j0} = x_{i,n-2} = 0 \qquad \text{iff } i \prec j \tag{30}$$

$$p_{ij} = 1 \qquad \text{iff } i \prec j, \tag{31}$$

$$p_{ij} \geq 0 \qquad \text{otherwise} \tag{32}$$

Constraints (26)-(28) link the y_{ij}^t and p_{ij} variables, prevent subtours and enforce the precedence relations. Redundant Constraints (29) are used to strengthen the formulation. These constraints are valid because if $i \prec j \prec k$ then $y_{ik}^t = 0$ since job j must be executed in between jobs i and k (Sarin et al., 2005). Redundant Constraints (30) follow a similar reasoning: if i needs to precede j, than i cannot be the last job in the sequence and j cannot be the first job.

The number of constraints in this model is cubic in the number of jobs (Constraint (28)). Preliminary computational experiments revealed that the model becomes inhibitively large for practical applications: just solving the LP relaxation of instances with less than 50 vertices could take 5-10 hours. Similar conclusions are drawn by Sarin et al. (2005). To improve the scalability of this model, we developed a simple separation routine for inequalities (28) (Algorithm 1). In this routine, we initialize the values p_{ij}^*, for all i, j, to the values attained by the p_{ij} variables in the LP relaxation. Each time a violated inequality is encountered for a given i, j, k, we add the following cut to the model:

$$p_{ij} + p_{jk} + p_{ki} + \sum_{t=1}^{n-1} y_{ji}^t \leq 2 \tag{33}$$

Cuts of the form (33) are shown to be valid and stronger than inequalities (28) (Sarin et al., 2005). Compared to the original work by Sarin et al. (2005), when solving the LP relaxation, this separation routine realizes an average speedup of one order of magnitude.

For the residual costs in the additive bounding, the reduced costs $\overline{y}_{ij}^t$, $\overline{x}_{ij}$ and $\overline{p}_{ij}$ for resp. the y_{ij}^t, x_{ij}, and p_{ij} variables are used. This time however, we

Algorithm 1: Separation of inequalities (28).

```
forall i, j : p*_ij > 2/3
    forall k = 1, ..., n − 1; k ≠ i ≠ j
        if p*_ij + p*_jk + p*_ki > 2 then
            generateCut()
            p*_ij ← 0                          (Prevents duplicate cuts)
```

cannot set $\Delta^t_{ij} = \overline{y}^t_{ij} + \overline{x}_{ij} + \overline{p}_{ij}$, because the costs $\overline{p}_{ij}$ must be counted for *every* job i that precedes job j in the solution. Instead we modify Equation (4) to:

$$\overline{s}(a) = \begin{cases} 0 & \text{if } u = \mathbf{r}, \\ \min\limits_{\substack{a' \in in(u), \\ val(a) \neq val(a')}} \{\overline{s}(a') + \Delta^{\ell(a')}_{val(a'),val(a)} + \sum\limits_{\substack{j \in All(v), \\ val(a) \neq j}} \overline{p}_{val(a),j}\} & \text{otherwise.} \end{cases} \quad (34)$$

where $\Delta^t_{ij} = \overline{y^t_{ij}} + \overline{x}_{ij}$. Notice that this modification is straightforward as all information required to make this calculation is already present in the MDD.

6. Computational Experiments

We now report our empirical findings on a set of TD-TSP, TD-TSP-TW, and TD-SOP instances. Experiments were conducted on an Intel Core I-7-4790 2.6GHz CPU, 16GB RAM, using CPLEX and CP Optimizer v. 12.6.3.

For the TD-TSP, TD-TSP-TW and TD-SOP, both Dantzig-Fulkerson-Johnson subtour elimination constraints, as well as 4-cycle inequalities (Equation (17)) are separated. For the separation of the 4-cycle inequalities we use the separation routine described in Miranda-Bront et al. (2014). At most ten rounds of separation are performed per node for TD-TSP(TW) instances, and at most four rounds for TD-SOP instances. At each round, all violated 4-cycle constraints as well as 30 of the most violated subtour inequalities are added to the model. Finally, it should be noted that no solutions for pure CP models relying solely on an `alldifferent` constraint have been included (Section 3), due to the low quality of these solutions. Hence, our primary comparison is based on the CP approach augmented with MDDs (CP_{MDD}), its extension with additive bounding (CP^{ab}_{MDD}), and the MIP models for each problem class. In addition, for the TD-TSP we compare with the approaches by Abeledo et al. (2013) and Miranda-Bront et al. (2010).

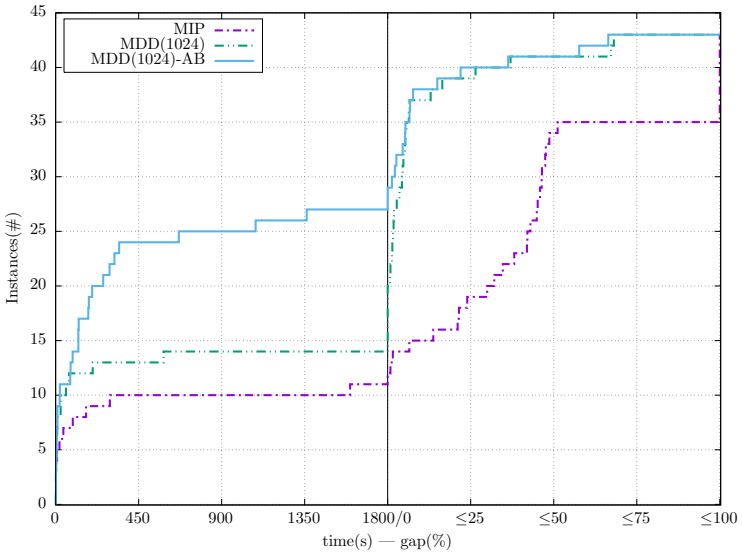
6.1. TDTSP

Analogous to Abeledo et al. (2013), the TD-TSP instances are derived from the TSPLib.[1] The setup times $\delta^t_{i,j}$ are defined as $(n-t)\dot{\delta}_{(i,j)}$, where $\dot{\delta}_{(i,j)}$ is the distance between cities i and j as specified in the TSPLib instance. For the TDTSP experiments, all instances reported in Abeledo et al. (2013) and Miranda-Bront et al. (2014) with less than 80 nodes have been used (43 instances in total). Detailed results for each of the 43 instances are available in Appendix A, Supplemental Material.

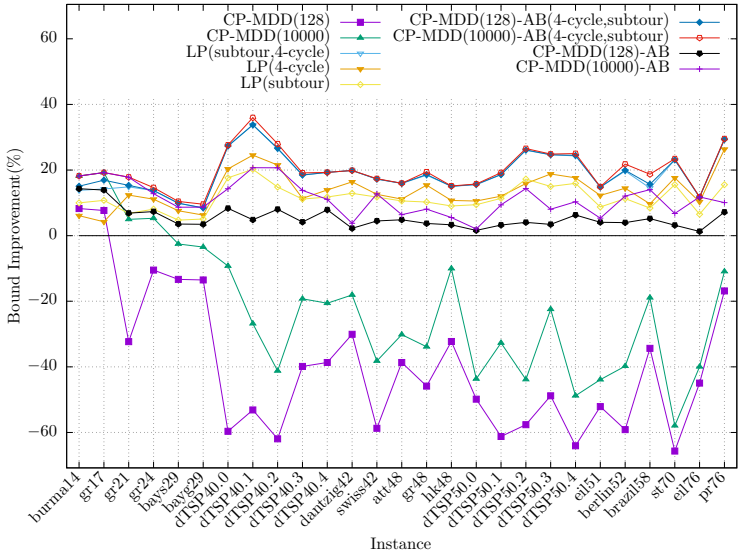
We first analyze the performance CP^{ab}_{MDD}, CP_{MDD}, and MIP: A performance plot comparing these methods is provided in Figure 4a. The plot is divided in two parts: the left half shows the number of instances that a method has solved within t seconds (the horizontal axis shows the time from 0 up to 1,800s). That is, each of these instances is solved within t seconds. To compare instances that were not solved to optimality, we depict on the right half the number of instances that have an optimality gap of at most $k\%$, for increasing values of k. The gap (percentage) for a particular instance is computed as: $100 \times \frac{UB-LB}{UB}$, where UB is a feasible solution, and LB the strongest available lower bound. More precisely, the LB is the strongest available bound: the optimal solution (whenever available), LP bound, CP^{ab}_{MDD} and CP_{MDD} root node bounds, or MIP bound. Note that some instances lie directly on the division line (1,800s/0% gap), indicating that an optimal solution was found, but optimality could not be proven within the time limit of 1,800 seconds. As can be observed from Figure 4a, both MDD approaches outperform MILP. A larger number of instances was solved within the time limit, and the optimality gap of the unsolved instances is smaller. Similarly, CP^{ab}_{MDD} outperforms CP_{MDD}. Despite the overhead introduced by the additive bounding procedure, CP^{ab}_{MDD} outperforms CP_{MDD} for instances up to 52 vertices. Beyond that point, the relaxation used in the experiments is either weak or requires excessive computational effort.

Miranda-Bront et al. (2010, 2014) compare the formulations proposed by Picard and Queyranne (1978) and Vander Wiel and Sahinidis (1995), analyzing the relationship between them and deriving valid inequalities and facets. Computational results are also presented for a branch-and-cut algorithm that uses these inequalities. The largest instance from Miranda-Bront et al. (2014) solved

[1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/



(a) Performance plot



(b) Bound comparison

Figure 4: TDTSP

to optimality is `eil51` (51 vertices) which took 90.55s. This is also the largest instance solved to optimality by our CP^{ab}_{MDD} approach. While our method can be regarded competitive with Miranda-Bront et al. (2010), both methods are generally outperformed by the approach of Abeledo et al. (2013), as shown in

Table 1, Appendix A, Supplemental Material.

Figure 4b provides a detailed comparison of the bounds derived from the various methods. For readability, the results are depicted for a subset of 27 out of 37 instances. As a base line, represented by the 0% line in Figure 4b, the TDTSP LP relaxation without extra inequalities is used. For each bounding procedure, Figure 4b shows how much stronger (or weaker) the derived bounds are, compared to the LP relaxation. When comparing the pure CP_{MDD} root node, it can be observed that the LP bound is generally stronger, except for very small instances. The larger the MDD width, the stronger the bound. When the MDD is strengthened with the simple LP relaxation, the resulting bound is consistently stronger than both the LP and pure MDD root node bounds for all instances.

Next, we investigate what happens if we strengthen the LP bound through valid inequalities, e.g., by adding (1) subtour inequalities, (2) 4-cycle inequalities, or (3) both. Regardless of which inequalities we add, the resulting LP relaxation (*LP(4-cycle)*, *LP(subtour)*, *LP(4-cycle,subtour)*) is consistently stronger than the bound derived from the MDD strengthened with the simple LP (*CP-MDD(·)-AB*). If we use the strengthened LP in the additive bounding procedure (*CP-MDD(·)-AB(4-cycle,subtour)*), we observe only a very small improvement over *LP(4-cycle,subtour)*. Interestingly, the *CP-MDD(·)-AB* yielded a much bigger improvement over both the *LP* and *CP-MDD(·)* approaches, than *CP-MDD(·)-AB(4-cycle,subtour)* yields over *LP(4-cycle,subtour)*. This phenomenon can be explained by the fact that the LP, strengthened with valid inequalities, captures a significant amount of problem structure. When this is projected onto the MDD, the resulting combined relaxation has approximately the same strength. In contrast, when a weaker relaxation is projected onto the MDD, the MDD adds more of the problem structure to the relaxation, resulting in a proportionally larger increase in strength of the relaxation.

6.2. TDTSP-TW

For the TDTSP-TW, 270 instances have been randomly generated following a procedure related to Dumas et al. (1995). Specifically, we fixed the number of vertices to $n \in \{30, 35, 40\}$, and generated asymmetrical travel distances for each vertex pair (i, j) drawing uniformly at random from the set $\{5, 100\}$. To ensure the instances were feasible, the time windows were set as follows. Once a distance matrix has been created, we would generate a permutation of cities uniformly at random, and compute the shortest travel distance to arrive at each

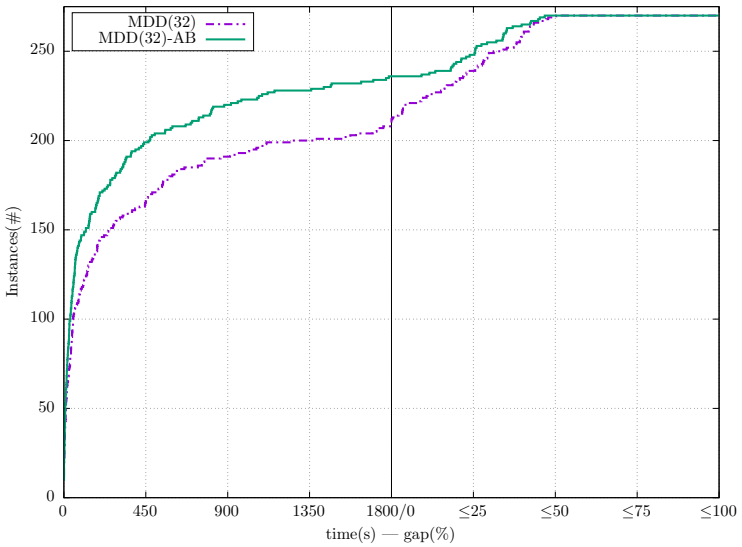
city according to the given permutation. For each city i in the permutation and the resulting arrival time t_i, the release date and deadlines were then drawn uniformly at random from $\{t_j, \ldots, t_i\}$ and $\{t_i, \ldots, t_{j'}\}$, respectively, where $j = \max\{0, i - w\}$ and $j' = \min\{n, i + w\}$, for some w (i.e., larger w have longer time windows, on average). For each $n \in \{30, 35, 40\}$, we considered $w \in \{10, \ldots, 90, 95\}$ and generated 5 different instance according to this procedure, resulting in 270 instances in total (90 per each n). Detailed results for each of the 270 instances are available in Appendix B, Supplemental Material.

Figures 5a, 5b compare the impact of the CP^{ab}_{MDD} and CP_{MDD} approaches. Lower bounds for these instances are obtained by taking the maximum over: LP, CP^{ab}_{MDD} root node, strongest MIP bound, optimal solutions for MIP, CP^{ab}_{MDD}, CP_{MDD}. Interestingly, MIP was unable to find a feasible solution for any of the instances. Hence, for clarity, a line for MIP has been omitted from Figures 5a, 5b. In contrast, both CP^{ab}_{MDD} and CP_{MDD} found feasible solutions for all instances. Furthermore, CP^{ab}_{MDD} significantly outperforms CP_{MDD} for these instances. Figures 5a and 5b also indicate that when the MDD width is small (Figure 5a), the Additive Bounding significantly strengthens the model. When the width of the MDD increases (Figure 5b), the impact of the Additive Bounding decreases, since the MDD is more expressive.

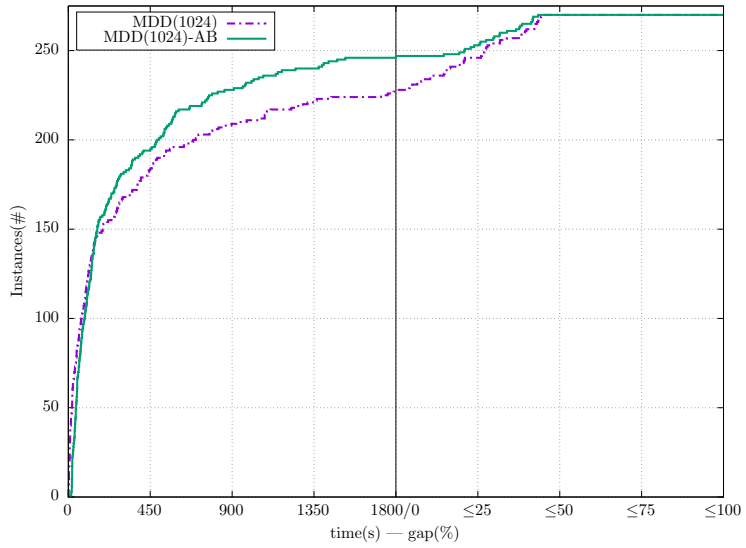
Figure 6 plots the performance of CP^{ab}_{MDD}, CP_{MDD} for the three different instance sizes: 30, 35 and 40 vertices. For the 30-vertex instances, the pure CP_{MDD} outperforms CP^{ab}_{MDD}. These instances are relatively easy, and as such, can be solved through the pure CP_{MDD} which does not suffer from the extra computational overhead present in the CP^{ab}_{MDD} approach. However, when the size of the instances increases, CP^{ab}_{MDD} clearly starts to outperform CP_{MDD}, as can be observed for both $n35$ and $n40$. When the instances continue to grow in size, the quality of the CP^{ab}_{MDD} solutions will gradually reduce to the point where there is no benefit of using the Additive Bounding procedure. This behavior can directly be explained by the fact that the quality of the LP relaxation used in the Additive Bounding procedure reduces when the instances become bigger. To resolve this issue, a stronger bounding procedure would be required.

6.3. TDSOP

For TDSOP, 29 instances with 7 to 100 vertices are derived from the SOP dataset in the TSPLib. The instances are transformed into their time-dependent equivalents in the same way as considered for the TD-TSP. Detailed results for each instance are available in Appendix C, Supplemental Material. Experiments



(a) MDD width 32



(b) MDD width 1024

Figure 5: Performance plot TDTSP-TW

revealed that it was too expensive to evaluate equation (4) during each filtering loop of the MDD. Instead equation (1) was used, thereby ignoring the reduced costs associated with the p_{ij} variables. The impact of this decision on the quality of the bounds turned out to be negligible, since the vast majority of reduced

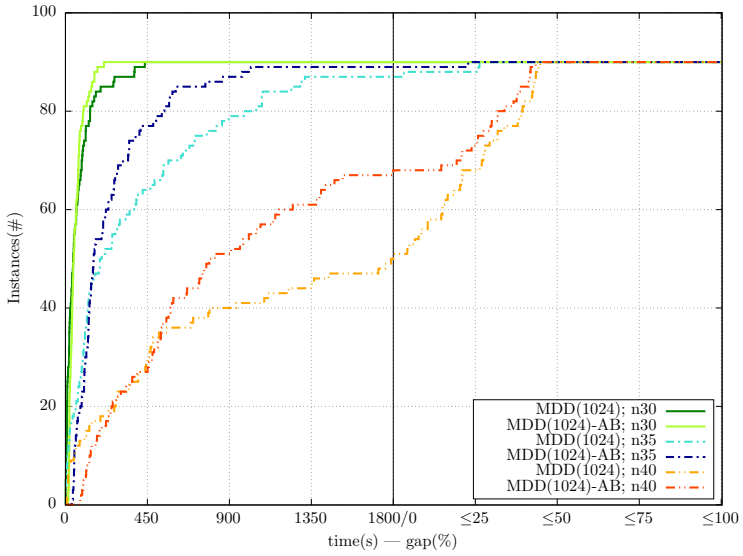
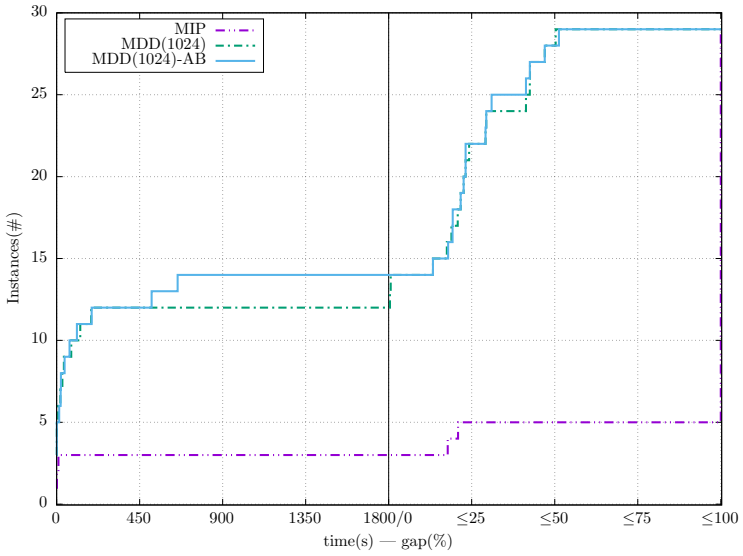


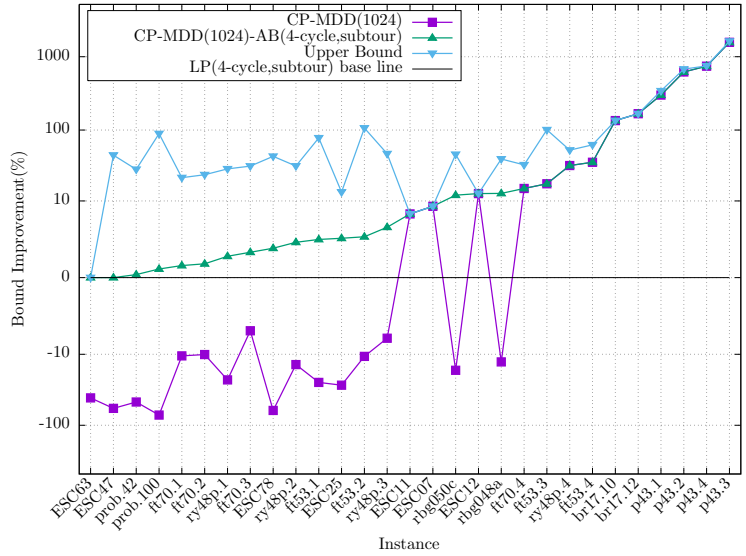
Figure 6: performance plot TDTSP-TW for different instance sizes

costs associated with the p_{ij} variables were equal to zero.

A performance plot is again provided in Figure 7a. Similar to the TDTSP and TDTSP-TW results, both MDD methods significantly outperform MIP. In fact, for most instances, the MILP model failed to find a single feasible solution. However, when comparing CP^{ab}_{MDD} and CP_{MDD} mutually, the differences are much less profound than in the foregoing experiments. These results are directly explained by a weak LP relaxation used in the AB procedure. Even though the model by Sarin et al. (2005) is known to provide strong bounds for the ATSP with precedence constraints, we do not obtain similar high-quality results for TDSOP. Moreover, even though we separate inequalities (28), we still had to limit the number of rounds of separation to four due to excessive computation times. For reference purposes, for an instance with 100 vertices and ten rounds of separation, computation times for the LP relaxation could easily exceed 24 hours. By limiting the number of rounds of separation, the quality of the LP bound reduced, and consequently the performance of CP^{ab}_{MDD} degraded. These findings seem to be in agreement with the conclusion of Sarin et al. (2005): 'While the relative tightness of this formulation does not generally translate to a competitive performance for ATSP problems without precedence relationships due to the size and structure of the LP relaxations, it is hoped that with advances



(a) Performance plot



(b) Bound comparison

Figure 7: TDSOP

in LP technology, it might prove to be more favorable in the future.' Indeed, instances in their experiments do not exceed 30 vertices.

Figure 7b compares the LP bound, and the CP_{MDD} and CP^{ab}_{MDD} root node bounds mutually. As a base line, the TDSOP LP relaxation strengthened with

all inequalities is used. The graph depicts how the root node bounds of CP_{MDD} and CP^{ab}_{MDD} compare against the LP bound: a positive and a negative value indicates that the bound is stronger and weaker, respectively. The "Upper Bound" line represents the maximum possible improvement over the LP bound, derived from the best primal solutions. To compensate for the large differences in y-values, we applied a log-modulus transformation ($L(x) = \text{sgn}(x) \times \log(|x| + 1)$) (John and Draper, 1980). For the first instances (left side of the graph), the CP_{MDD} root node is significantly weaker than the LP relaxation. Nevertheless, through the Additive Bounding procedure, the root node can be strengthened significantly. In fact, for a number of instances, the CP^{ab}_{MDD} root node bound gets very close to the optimal solution. Finally, for instances where the LP bound is really weak, the root node bounds of CP_{MDD} and CP^{ab}_{MDD} coincide.

7. Conclusion

In this work, the TDTSP(-TW) and TDSOP problems have been solved through a novel hybrid approach that combines constraint programming, linear programming, and MDDs. Since CP models are often ineffective in solving sequencing problems like the TSP, MDDs are incorporated to strengthen the model. By integrating the MDDs, significantly more information about the problem structure and solution space are consolidated into the CP model. Furthermore, since bounds on the optimal solution can be computed through the MDDs, domain propagation is improved, the search tree can be pruned more effectively, and computational effort required to prove optimality is reduced. Finally we show how structural information from different problem relaxations can be incorporated into the MDD through additive bounding.

Computational experiments clearly show that the hybrid approach outperforms traditional MILP and CP formulations for both sequencing problems in terms of time and solution quality. For a number of instances, the traditional CP approach was able to find the optimal solution, but failed to prove optimality in a reasonable amount of time. Due to the bounds calculated by the MDD propagator, the integrated approach could attest optimality for these instances orders of magnitude faster.

The approach presented in this work is generic in the sense that it hardly relies on problem-specific information. Therefore, future research could be aimed at expanding this work to related sequencing problems. Alternatively, one could

attempt to include existing dedicated solution approaches for the TD-TSP and TD-SOP into the current framework to improve its overall performance.

Bibliography

H. G. Abeledo, R. Fukasawa, A. A. Pessoa, and E. Uchoa, "The time dependent traveling salesman problem: polyhedra and algorithm," *Math. Program. Comput.*, vol. 5, no. 1, pp. 27–55, 2013.

J. Albiach, J. M. Sanchis, and D. Soler, "An asymmetric TSP with time windows and with time-dependent travel times and costs: An exact solution through a graph transformation," *European Journal of Operational Research*, vol. 189, no. 3, pp. 789 – 802, 2008.

H. R. Andersen, T. Hadzic, J. N. Hooker, and P. Tiedemann, "A constraint store based on multivalued decision diagrams," in *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming*, ser. CP'07. Springer-Verlag, 2007, pp. 118–132.

D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, *The Traveling Salesman Problem.* Princeton University Press, 2006.

D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, 2006.

S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of robots by learning," *Journal of Robotic Systems*, vol. 1, no. 2, pp. 123–140, 1984.

R. Baldacci, A. Mingozzi, and R. Roberti, "New state-space relaxations for solving the traveling salesman problem with time windows," *INFORMS Journal on Computing*, vol. 24, no. 3, pp. 356–371, 2012.

P. Baptiste, P. Laborie, C. L. Pape, and W. Nuijten, *Handbook of Constraint Programming*, 1st ed. Elsevier, 2006, ch. 22: Constraint-Based Scheduling and Planning, pp. 761–799.

P. Benchimol, W.-J. v. Hoeve, J.-C. Régin, L.-M. Rousseau, and M. Rueher, "Improved filtering for weighted circuit constraints," *Constraints*, vol. 17(3), pp. 205–233, 2012.

D. Biskup, "Single-machine scheduling with learning considerations," *European Journal of Operational Research*, vol. 115, no. 1, pp. 173 – 178, 1999.

A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan, "The minimum latency problem," in *Proceedings of the Twenty-sixth Annual ACM*

Symposium on Theory of Computing, ser. STOC '94. New York, NY, USA: ACM, 1994, pp. 163–171.

A. Cire and W. J. van Hoeve, "Multivalued decision diagrams for sequencing problems," *Operations Research*, vol. 61, no. 6, pp. 1411–1428, 2013.

W. Cook, *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton University Press, 2012.

J.-F. Cordeau, G. Ghiani, and E. Guerriero, "Analysis and branch-and-cut algorithm for the time-dependent travelling salesman problem," *Transportation Science*, vol. 48, no. 1, pp. 46–58, 2012.

M. Desrochers and G. Laporte, "Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints," *Operations Research Letters*, vol. 10, no. 1, pp. 27 – 36, 1991.

Y. Dumas, J. Desrosiers, E. Gelinas, and M. M. Solomon, "An optimal algorithm for the traveling salesman problem with time windows," *Operations Research*, vol. 43, no. 2, pp. 367–371, 1995.

M. Fischetti and P. Toth, "An additive bounding procedure for combinatorial optimization problems," *Operations Research*, vol. 37, no. 2, pp. 319–328, 1989.

M. Fischetti, G. Laporte, and S. Martello, "The delivery man problem and cumulative matroids," *Oper. Res.*, vol. 41, no. 6, pp. 1055–1064, Nov. 1993.

S. French, *Sequencing and Scheduling*. John Wiley & Sons, 1982.

M. R. Garey and D. S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

M. Gendreau, G. Ghiani, and E. Guerriero, "Time-dependent routing problems: A review," *Computers & Operations Research*, vol. 64, pp. 189 – 197, 2015.

M. T. Godinho, L. Gouveia, and P. Pesneau, "Natural and extended formulations for the time-dependent traveling salesman problem," *Discrete Applied Mathematics*, vol. 164, Part 1, pp. 138 – 153, 2014.

L. Gouveia and M. Ruthmair, "Load-dependent and precedence-based models for pickup and delivery problems," *Computers & Operations Research*, vol. 63, pp. 56 – 71, 2015.

L. Gouveia and S. Voss, "A classification of formulations for the (time-dependent) traveling salesman problem," *European Journal of Operational Research*, vol. 83, no. 1, pp. 69 – 82, 1995.

G. Heilporn, J.-F. Cordeau, and G. Laporte, "The delivery man problem with time windows," *Discrete Optimization*, vol. 7, no. 4, pp. 269 – 282, 2010.

M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees," *Operations Research*, vol. 18, no. 6, pp. 1138–1162, 1970.

J. N. Hooker, *Integrated Methods for Optimization (International Series in Operations Research & Management Science)*, 2nd ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2012.

X. Huang and J.-J. Wang, "Machine scheduling problems with a position-dependent deterioration," *Applied Math. Modelling*, vol. 39, no. 1011, pp. 2897 – 2908, 2015.

A. A. K. Jeng and B. M. T. Lin, "Makespan minimization in single-machine scheduling with step-deterioration of processing times," *The Journal of the Operational Research Society*, vol. 55, no. 3, pp. pp. 247–256, 2004.

J. A. John and N. R. Draper, "An alternative family of transformations," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 29, no. 2, pp. 190–197, 1980.

A. B. Keha, K. Khowala, and J. W. Fowler, "Mixed integer programming formulations for single machine scheduling problems," *Computers & Industrial Engineering*, vol. 56, no. 1, pp. 357 – 367, 2009.

A. Lodi, M. Milano, and L.-M. Rousseau, "Discrepancy-based additive bounding procedures," *INFORMS J. on Computing*, vol. 18, no. 4, pp. 480–493, Jan. 2006.

A. Lucena, "Time-dependent traveling salesman problem–the deliveryman case," *Networks*, vol. 20, no. 6, pp. 753–763, 1990.

C. Malandraki and M. Daskin, "Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms," *Transportation Science*, vol. 26, no. 3, pp. 185–200, 1992.

I. Méndez-Díaz, P. Zabala, and A. Lucena, "A new formulation for the traveling deliveryman problem," *Discrete Appl. Math.*, vol. 156, no. 17, pp. 3223–3237, 2008.

J. J. Miranda-Bront, I. Méndez-Díaz, and P. Zabala, "An integer programming approach for the time-dependent tsp," *Electronic Notes in Discrete Mathematics*, vol. 36, pp. 351–358, 2010.

J. J. Miranda-Bront, I. Mndez-Daz, and P. Zabala, "Facets and valid inequalities for the time-dependent travelling salesman problem," *European Journal of Operational Research*, vol. 236, no. 3, pp. 891 – 902, 2014.

J. C. Picard and M. Queyranne, "The Time-dependent Traveling Salesman Problem and its Application to the Tardiness Problem in One-machine Scheduling," *Operations Research*, vol. 26, no. 1, pp. 86–110, 1978.

M. Pinedo, *Scheduling: Theory, Algorithms and Systems*, 3rd ed. Prentice Hall, 2008.

M. Queyranne, M. Queyranne, and A. S. Schulz, "Polyhedral approaches to machine scheduling," Technical University of Berlin, Department of Mathematics, Berlin, Germany, Tech. Rep. 408/1994, 1994.

J.-C. Régin, "A Filtering Algorithm for Constraints of Difference in CSPs," in *Proceedings of AAAI*, vol. 1. AAAI Press, 1994, pp. 362–367.

J.-C. Régin, "Global constraints: A survey," in *Hybrid Optimization*, ser. Springer Optimization and Its Applications, P. van Hentenryck and M. Milano, Eds. Springer New York, 2011, vol. 45, pp. 63–134.

R. Rudek, "Scheduling problems with position dependent job processing times: computational complexity results," *Annals of Operations Research*, vol. 196, no. 1, pp. 491–516, 2012.

S. C. Sarin, H. D. Sherali, and A. Bhootra, "New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints," *Operations Research Letters*, vol. 33, no. 1, pp. 62 – 70, 2005.

D. Simchi-Levi and O. Berman, "Minimizing the total flow time of n jobs on a network," *IIE Transactions*, vol. 23, no. 3, pp. 236–244, 1991.

J. van den Akker, C. van Hoesel, and M. Savelsbergh, "A polyhedral approach to single-machine scheduling problems," *Mathematical Programming*, vol. 85, no. 3, pp. 541–572, 1999.

C. van Eijl, "A polyhedral approach to the delivery man problem," 1995.

R. J. Vander Wiel and N. V. Sahinidis, "Heuristic bounds and test problem generation for the time-dependent traveling salesman problem," *Transportation Science*, vol. 29, no. 2, pp. 167–183, 1995.

C.-C. Wu, W.-C. Lee, and T. Chen, "Heuristic algorithms for solving the maximum lateness scheduling problem with learning considerations," *Computers & Industrial Engineering*, vol. 52, no. 1, pp. 124 – 132, 2007.

Y. Yin, D. Xu, K. Sun, and H. Li, "Some scheduling problems with general position-dependent and time-dependent learning effects," *Information Sciences*, vol. 179, no. 14, pp. 2416 – 2425, 2009, including Special Section Linguistic Decision Making Tools and Applications.

Y. Yin, M. Liu, J. Hao, and M. Zhou, "Single-machine scheduling with job-position-dependent learning and time-dependent deterioration," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 42, no. 1, pp. 192–200, Jan 2012.