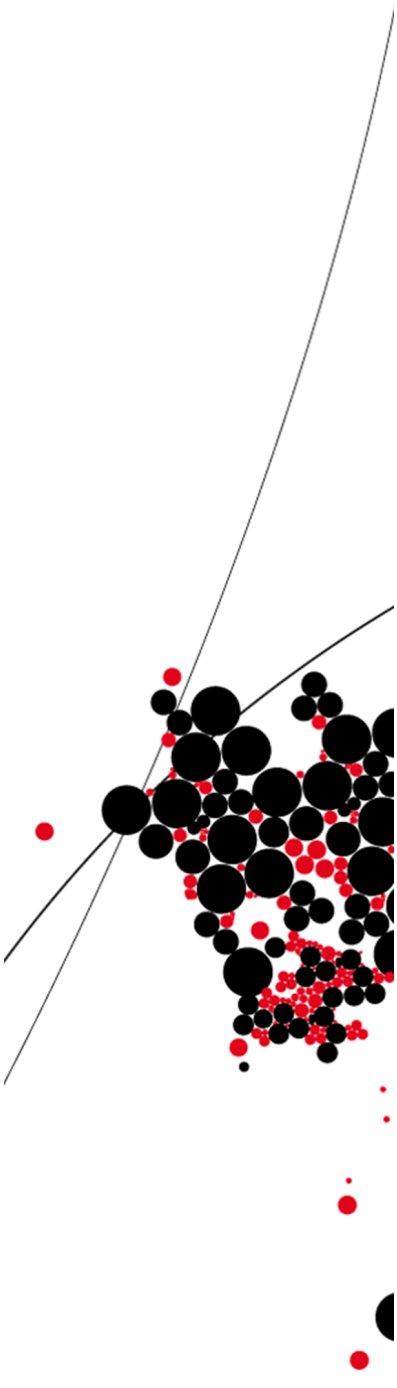
UNIVERSITY OF TWENTE.

**Faculty of Electrical Engineering,
Mathematics & Computer Science**

Recognition and Exploitation of Single-Machine Scheduling Subproblems in Mixed Integer Programs

Reinout Lambertus Henricus Wijfjes

M.Sc. Thesis
January 2022



Supervisors:
prof. dr. M. J. Uetz
dr. M. Walter

Graduation committee:
prof. dr. M. J. Uetz
dr. M. Walter
dr. J. C. W. van Ommeren
dr. H. Hang

Discrete Mathematics and
Mathematical Programming
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Abstract

To speed up the branch-and-bound algorithm that is used to solve mixed integer programs, modern solvers exploit general and problem specific cutting planes. As it is unknown whether the application of the class of scheduling cutting planes is beneficial, we investigate the employment of the subclass of single machine scheduling cutting planes that are valid for the formulation using big-M constraints with natural date variables. To be able to identify single machine scheduling subproblems within mixed integer programs, a new exponential time recognition algorithm, performing well in practice, is developed and presented here. Experiments to investigate the exploitation of single machine scheduling subproblems indicate that in a small number of cases the use of scheduling cutting planes indeed reduce the solving time and in a few more cases there is potential to decrease the solving time by integrating the cutting planes into the branch-and-cut algorithm.

Preface

In front of you lies the master thesis *Recognition and Exploitation of Single-Machine Scheduling Subproblems in Mixed Integer Programs*, that marks the end of year of work on my final project for the master Applied Mathematics at the University of Twente. As this thesis was written during the COVID19-pandemic, the process that lead to this thesis can be described as a bumpy road. More often than not I had to work in my student room and this has not always been easy. Therefore, I want to take the opportunity to thank everyone who helped and supported me.

First of all, I want to thank my supervisors dr. Matthias Walter and prof. dr. Marc Uetz. Thank you for guidance, input and feedback. Maybe even more important, thank you for the energy I got from our meetings. Many times I came into one of our meetings feeling like my progress was not optimal, but your understanding when I struggled, your kind words, enthusiasm and positive feedback always lifted my spirits to continue working on my final project. Furthermore, I want to thank the other members of my graduation committee, dr. Jan-Kees van Ommeren en dr. Hanyuan Hang.

Additionally, I want to thank everybody with whom I had the pleasure to study together, online or on campus. An honorable mention is for my fellow members of the EEMCS Graduation Support Group. Our weekly meetings on Monday helped me keeping an overview of my progress and motivated me to keep on moving forward with my thesis. Another special mention is for Rolf, whose offer to provide feedback on this thesis I gladly accepted. Moreover, I want to thank my roommates at Huize Frits is Dood and my fellow members of D.B.V. DIOK, who where there when I needed relaxation.

Finally, I want to thank my family and friends, on who I always can resort whenever I want.

Contents

1 Introduction

1.1 The Mixed Integer Programming model

How should a farmer allocate his available resources (e.g. machinery, land, seeds, fertilizer) to maximize his total food production? What is the design of a telecommunication network that guarantees phone range in the whole of the Netherlands against minimal costs? How should buses and bus drivers be assigned to different routes such that the public transport timetable is met? An answer to these kind of questions can be found using a powerful modelling tool from the world of mathematics: mixed integer linear programming (MIP).

In a MIP model, the modeller sets a linear objective function that needs to be optimized. The variables are subject to a set of m linear constraints. Besides that, some of the variables are restricted to be binary, while other variables can take any integer or any real value. This is captured in the index sets $B \quad I \quad R$, where B contains indices of binary variables, I contains indices of integer variables and $R = \{1, \ldots, r\}$ contains all variable indices. These index sets are given and form the input of the problem, together with the objective function and constraint set. This can be summarized in the following general form of a MIP:

$$\min \quad c^T x \tag{1a}$$

$$\text{s.t.} \quad Ax \quad b, \tag{1b}$$

$$x_i \in \{0, 1\}, \qquad \forall i \in B, \tag{1c}$$

$$x_i \in \mathbb{Z}, \qquad \forall i \in I \cap B, \tag{1d}$$

$$x_i \in \mathbb{R}, \qquad \forall i \in R \cap I, \tag{1e}$$

with matrix $A \in \mathbb{R}^{m \quad r}$, vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^r$. Furthermore, Q is the index set of the constraints.

1.2 Techniques for solving Mixed Integer Programs

Modern MIP solvers use a lot of methods and tricks to solve models as quickly as possible. The foundation of solver software is formed by the branch-and-bound (B&B) algorithm, which is described here first.

The B&B method, as introduced by Land and Doig in 1960 [1], solves the optimization problem by starting with dropping the integrality constraints (1c) and (1d). Then, the set of feasible solutions of the obtained LP is called the *linear programming (LP) relaxation*. This LP relaxation contains all feasible solutions of the original MIP, but can be solved to optimality relatively easy, for example by the simplex algorithm. The found optimal solution is checked for integer feasibility: do the variables that in the original model should take an integer value, indeed take an integer value in this optimal solution? If so, the optimal solution of the original problem is found. If not, the objective value of the optimal solution is set to be the *dual bound*. In that case, also an integer

variable x_i, taking non-integral value x_i in the optimal solution, is chosen to *branch* on. Branching happens by creating new MIP models. As an example, it is possible to create two new models. The first model is a copy of the original model with the additional constraint $x_i \quad \lfloor x_i \rfloor$, while the second model is a copy of the original with the additional constraint $x_i \quad \lceil x_i \rceil$. The new models are marked as *open*, while the solved model is marked as *closed*. The term branching comes from the fact that the the original model can be viewed as the root node of a tree, while the new models are the branches/leafs of the tree.

After this initialization, the algorithm continues in a loop. In one iteration of this loop, the algorithm selects one of the open nodes of the B&B-tree. Once again, the LP relaxation of the MIP corresponding to the node is solved to optimality. A few things then can happen. First, the found solution could be an integer feasible solution. If the objective value is better than the objective value of the then best-known solution, the *incumbent* and the *primal bound* are updated accordingly. Here, the incumbent is the best-known feasible solution and the primal bound is the objective value of the incumbent. After the update, the node is marked as closed, but this also happens when the feasible solution does not yield an improvement. When the found solution is not integer feasible, branching takes place, creating new open nodes and closing the current node. Finally, the constraints that are added to the MIP in a previous branching step may have caused infeasibility of the MIP instance corresponding to a node in the B&B-tree. In that case, the current node is closed without branching. The algorithm terminates when all nodes are closed or when the primal and dual bound have the same value.

To speed up the B&B algorithm, MIP solvers use all kinds of tricks. One of those tricks is preprocessing: before operating the B&B algorithm, solvers could for example try to simplify constraints, remove redundant constraints or reduce the number of variables or constraints by substitution. Furthermore, MIP solvers use heuristics to find feasible solutions of the MIP, that hopefully help improving the incumbent and primal bound. Both preprocessing and heuristics will not be discussed here in further detail. Some other influences on the performance of the B&B algorithm are the algorithm that decide which open node is processed next or the rule on how to perform branching.

Another trick of MIP solvers to prove faster that the solution is optimal is the use of *cutting planes*. To explain the concept of cutting planes (or cuts), suppose the B&B algorithm has found an optimal solution for the LP relaxation of some MIP instance. Suppose furthermore that this solution is not integer feasible. Then a cutting plane is an inequality that is valid for all integer feasible solutions, but invalid for the optimal solution of the LP relaxation. If such an inequality can be found, it can be added to the MIP, after which the LP relaxation of the MIP can be solved again. This process could be repeated multiple times in every node of the B&B tree and is known as the *separation problem*. A drawback of repeating this method often without finding the optimal solution, is that it gets harder to solve the LP relaxation because of the growing number of constraints. Therefore, MIP solvers will spend a limited amount of time looking for cutting planes per node of the B&B-tree. MIP solvers use many

different types of cuts, for example Chvátal-Gomory cuts, lift-and-project cuts, cover cuts and clique cuts [2], [3].

Besides these general cuts, MIP solvers can also use problem specific cutting planes for (sub)problems that are present in MIPs, such as the travelling salesman problem (TSP). Dantzig, Fulkerson and Johnson [4] showed that the TSP structure could be used to help solving MIPs faster. This shows that the exploitation of subproblems of a MIP is a useful trick to improve the solving time of MIPs. When the use of general and problem specific cutting planes is integrated in the B&B-algorithm, the algorithm is called branch-and-cut.

1.3 Research Question and Approach

A type of cutting planes for which it is not known whether exploitation decreases the solving time of a MIP are the scheduling cuts. The class of scheduling problems (and the corresponding cutting planes) consists of many different problems, such as single machine scheduling, parallel machine scheduling and job-shop problems. Because of time constraints and as the single machine scheduling problem is the simplest problem in this class, in this thesis only the single machine scheduling problem and the corresponding cutting planes will be considered. This leads to the following research question.

Research Question. *Can the presence of single machine scheduling subproblems in mixed integer programs be exploited to reduce the solving time of mixed integer programs?*

To be able to answer this research question, first various single machine scheduling MIP model are compared with each other. This comparison is used to pick one of the models to perform further research on. After this choice, the next step is to develop and test an algorithm that can recognize a single machine scheduling subproblem (SMSSP) in any MIP instance. This is necessary to be able to compute cutting planes that might help speeding up the B&B algorithm. As many valid inequalities for single machine scheduling are known, a selection of valid inequalities is chosen that will serve as cutting planes. Finally, experiments are performed to find out whether the selected single machine scheduling cuts help reducing the solving time of MIP instances in which a SMSSP is present.

1.4 Outline

In Section 2, the different single machine scheduling models are introduced and discussed, after which one of the models is chosen for further research. After this, an algorithm to recognize SMSSPs is presented in Section 3. Subsequently, the selection of cutting planes is presented in Section 4.

In Section 5, the results of the test of the recognition algorithm are given and analyzed. Also in this Section, the method and results of the experiments to discover whether the cutting planes help reducing the solving time of MIPs are provided together with its analysis. Finally, the recognition algorithm and the experiments on the solving time are discussed in Section 6.

2 MIP models for single machine scheduling

As described in the introduction, the goal of this research is to recognize a SMSSP in a MIP model and then exploit the presence of this subproblem to solve the MIP model faster. Therefore, it is necessary to know how such a problem is formulated as MIP model. In this section, several MIP models for single machine scheduling are introduced. Then, the formulations are compared with each other, after which is chosen which of those models will be used for recognition and exploitation.

Before the formulations can be given, it should first be clear which problem is modelled. In this study, the focus lies on the problem where n jobs need to be processed by one machine/server and the processing of the jobs should not be interrupted, i.e. preemption is not allowed. For every job $j \in J := \{1, 2, \ldots, n\}$, the possible parameters and variables are given in Table 1. Possible parameters are the processing time p_j, release date r_j, due date d_j and weight w_j, while possible variables are the completion time variables C_j and starting time variables S_j.

Table 1: Possible parameters and variables for every job $j \in J$.

Name	Type	Description
p_j	Parameter	Time required to process job j
r_j	Parameter	Earliest possible moment to start processing job j
d_j	Parameter	Latest possible moment to complete processing job j
w_j	Parameter	Represents the importance of job j
C_j	Variable	Moment of completing the processing of job j
S_j	Variable	Moment of starting the processing of job j

For the purpose of formulating different single machine scheduling MIP models, it is assumed that $r_j = 0, d_j = 1$ holds true for all jobs $j \in J$. Although not of relevance, but for the sake of completeness, in the following formulations it is assumed that the objective is to minimize the sum of weighted completion times, i.e. minimize $\sum_{j=1}^{n} w_j C_j$. To meet this objective, solving a MIP model would not be the most efficient method, as Smith has shown that the weighted shortest processing time algorithm solves this problem to optimality in $O(n \log n)$ time [5].

2.1 Natural date variables

The first single machine scheduling MIP model considered here is the variant with completion time variables C_j and binary ordering variables y_{jk}. The binary variable y_{jk} equals 1 if job j is scheduled before job k and 0 otherwise. The MIP that minimizes the total weighted completion time can be represented as

below:

$$\min \quad \sum_{j=1}^{n} w_j C_j \tag{2a}$$

$$\text{s.t.} \quad C_j + p_k \quad C_k + T(1 \quad y_{jk}), \qquad 8j,k \; 2 \; J : j \; 6= \; k, \tag{2b}$$

$$C_k + p_j \quad C_j + T y_{jk}, \qquad 8j,k \; 2 \; J : j \; 6= \; k, \tag{2c}$$

$$C_j \quad p_j, \qquad 8j \; 2 \; J, \tag{2d}$$

$$y_{jk} \; 2 \; f0,1g, \qquad 8j,k \; 2 \; J : j \; 6= \; k, \tag{2e}$$

$$C_j \; 2 \; \mathsf{R} \; _0, \qquad 8j \; 2 \; J, \tag{2f}$$

with $T \quad \sum_{j=1}^{n} p_j$.

In this model, constraint sets (2b) and (2c) are big-M constraints and guarantee that no pair of jobs can overlap, i.e. when job j is processed these constraints ensure that job k can not be processed at the same time. Constraint set (2d) implies that the completion time variables takes a value that is at least the processing time, while constraint sets (2e) and (2f) define the ordering variables and completion time variables, respectively.

Another variant of a model using natural date variables is to use starting time variables S_j instead of completion time variables C_j. In that case, constraint set (2d) is left out, C_j, C_k are replaced by S_j, S_k in constraint sets (2b), (2c) and (2f) and in constraint sets (2b and (2c) p_k is substituted by p_j and vice versa.

The first work on formulating a scheduling problem using natural date variables with disjunctive constraints was performed by Balas [6]. This formulation was later also studied by Queyranne [7] and Queyranne and Wang [8] and is also referred to as disjunctive constraints formulation or big-M formulation.

2.2 Time-indexed variables

In the model using time-indexed variables, the most important concept is a planning horizon discretized in a finite number of time periods $t = 1, 2, \ldots, T$, with $T \quad \sum_{j=1}^{n} p_j$ and where time period t starts at time $t \quad 1$ and ends at time t. Furthermore, the binary time index variable x_{jt} is introduced. This variable takes value 1 when the processing of job j starts at time t and otherwise equals 0. Then, the MIP that uses time-indexed variables to minimize the total weighted

completion time can be formulated as follows:

$$\min \quad \sum_{j=1}^{n} \sum_{t=1}^{T} \sum_{p_j+1}^{} w_j(t \quad 1 + p_j)x_{jt} \tag{3a}$$

$$\text{s.t.} \quad \sum_{t=1}^{T} \sum_{p_j+1}^{} x_{jt} = 1, \qquad 8j \; 2 \; J, \tag{3b}$$

$$\sum_{j=1}^{n} \sum_{s=\max(0,t}^{t} \sum_{p_j+1)}^{} x_{js} \quad 1, \qquad 8t \; 2 \; f1, 2, \ldots, Tg, \tag{3c}$$

$$x_{jt} \; 2 \; f0, 1g, \qquad 8j \; 2 \; J, 8t \; 2 \; f1, 2, \ldots, Tg. \tag{3d}$$

Using the variables of the time-indexed formulation, one can compute the completion time of job j as $C_j = \sum_{t=1}^{T} \sum_{p_j+1}^{} (t \quad 1 + p_j)x_{jt}$. Then, substituting this in the objective of the natural dates formulation (2a), the objective of the time-indexed formulation follows easily. Moreover, constraint set (3b) indicates that a job can start at only one moment, while constraint set (3c) assures that at every moment only one job can be processed. Constraint set (3d) specifies the binary time index variables.

The time-indexed variables formulation is introduced by Sousa and Wolsey [9]. Also van den Akker et al. [10] and Šorić [11] contributed in the research on this model.

2.3 Linear ordering variables

The linear ordering variables formulation uses only one type of variables: binary ordering variables δ_{jk}. This variable takes value 1 when job j is scheduled before job k and 0 otherwise. Using only these variables, one can model the single machine scheduling in the following way:

$$\min \quad \sum_{j=1}^{n} \left(w_j p_j + \sum_{k=1, k \neq j}^{n} w_j p_k \delta_{kj} \right) \tag{4a}$$

$$\text{s.t.} \quad \delta_{jk} + \delta_{kj} = 1, \qquad 8j, k \; 2 \; J, j \neq k, \tag{4b}$$

$$\delta_{jk} + \delta_{kl} + \delta_{lj} \quad 2, \qquad 8j, k, l \; 2 \; J, j \neq k \neq l, \tag{4c}$$

$$\delta_{jk} \; 2 \; f0, 1g, \qquad 8j, k \; 2 \; J, j \neq k. \tag{4d}$$

Using the linear ordering variables, the completion time of job j can be calculated through $C_j = p_j + \sum_{k=1, k \neq j}^{n} p_k \delta_{kj}$. After substituting this into the objective of the natural date variables formulation (2a), the objective of the linear ordering variables formulation (4a) follows after some rewriting. Additionally, constraint set (4b) imposes the restriction that jobs j and k can be ordered in only one way. Likewise, constraint set (4c) is the transitivity constraint set that implies linear ordering between every triplet of jobs, while constraint set (4d) sets the linear ordering variables.

The MIP formulation using linear ordering variables was initially presented by Dyer and Wolsey [12], while further research was performed by Blazewicz et al. [13], Nemhauser and Savelsbergh [14] and Chudak and Hochbaum [15].

2.4 Assignment and positional date variables

The idea behind this formulation is to assign the n jobs to n positional variables. Therefore, the binary assignment variables u_{jk} are defined such that it takes value 1 when job j is assigned to position k and 0 otherwise. Furthermore, this model uses completion time variables C_j and positional completion time variables γ_k. With these variables, the single machine scheduling problem can be modelled in the following manner:

$$\text{minimize} \quad \sum_{j=1}^{n} w_j C_j \tag{5a}$$

$$\text{subject to} \quad \sum_{k=1}^{n} u_{jk} = 1, \qquad \forall j \in J, \tag{5b}$$

$$\sum_{j=1}^{n} u_{jk} = 1, \qquad \forall k \in J, \tag{5c}$$

$$\gamma_1 \quad \sum_{j=1}^{n} p_j u_{j1}, \tag{5d}$$

$$\gamma_k \quad \gamma_{k\ 1} + \sum_{j=1}^{n} p_j u_{jk}, \qquad \forall k \in J \setminus \{1\}, \tag{5e}$$

$$C_j \quad \gamma_k \quad T(1 \quad u_{jk}), \qquad \forall j,k \in J, \tag{5f}$$

$$u_{jk} \in \{0,1\}, \qquad \forall j,k \in J, \tag{5g}$$

$$C_j, \gamma_k \in \mathbb{R}_{\ 0}, \qquad \forall j,k \in J, \tag{5h}$$

with $T \quad \sum_{j=1}^{n} p_j$.

In this formulation, constraint sets (5b) and (5c) ensure that every job is assigned to exactly one position and that every position gets assigned exactly one job. Constraint (5d) and constraint set (5e) are necessary to determine the positional completion time, while constraint set (5f) is essential for finding the completion times of jobs. Finally, constraint sets (5g) and (5h) define the binary assignment variables and the (positional) completion time variables.

Lasserre and Queyranne [16] came up with the assignment and positional dates variables formulation. This model is further explored by Dauzère-Pérès [17] and Sevaux and Dauzère-Pérès [18].

2.5 Comparison and choice of model

In order to pick a single machine scheduling model for recognition and exploitation, it is important to know what the computational performance of the

different models is. Several computational studies regarding those formulations have been performed. Already in 2005, Khowala et al. [19] found that for the objective of minimizing the total weighted tardiness, the LP relaxation of the formulation using assignment and positional date variables can be solved relatively quickly and therefore offers some promise for new advanced techniques. However, the formulations using time-indexed variables and linear ordering variables were preferred, due to the fact that they generally produce tighter bounds. On the other hand, the LP relaxations of these two formulations are known to be more difficult to solve. This means that fewer nodes in a B&B-tree can be explored in the same amount of time. In general, it is known that the formulation with natural date variables does not perform well. This is because of the presence of disjunctive big-M constraints. It is widely acknowledged that models containing those constraints often have a rather weak LP relaxation and solving larger models can be very difficult and time consuming.

Later studies, by Keha et al. [20] and Ying et al. [21], indeed show that the formulation using assignment and positional date variables outperforms the other formulations. Therefore, it makes more sense to look into the other formulations to see whether recognition and exploitation could be useful. All of the remaining formulations have disadvantages concerning the LP relaxation, but the formulations with time-indexed variables and linear ordering variables often have tight bounds as opposed to the the formulation with natural date variables. Loose bounds often means that more steps need to be taken to close the gap between the primal and dual bound. In this case that would mean that more nodes need to be explored in a B&B-tree, which clearly implies that it takes more time to prove that a solution is optimal. From this reasoning, it follows that it is more likely that progress can be made with the model that uses natural dates variables. So, performing research on this model offers the highest chance of formulating a positive answer to the research question. Therefore, the formulation with natural date variables is the model that will be used in the process of recognition and exploitation.

3 Recognizing a single machine scheduling subproblem

After the choice for the natural dates formulation, the next step is to find an algorithm that can recognize a SMSSP that uses the natural dates formulation. Before the algorithm can be given, first it is convenient to see what the algorithm gets as input. Accordingly, the constraints of the natural date formulation are reformulated into general form constraints, such that constraints are in a form as found in an arbitrary MIP model. Subsequently, the constraint graphs are defined, after which the algorithm is given. Finally, it will be shown that the algorithm indeed recognizes a SMSSP and what the complexity of the algorithm is.

3.1 Reformulating the single machine scheduling problem

Before the single machine scheduling problem is reformulated, it is useful to briefly repeat the general MIP model and the natural dates formulation of the single machine scheduling problem here first. For more detailed information about these formulations, see Sections 1.1 and 2.1.

The general MIP model uses variables x and is given below:

$$\begin{aligned}
\min \quad & c^T x && && (1a)\\
\text{s.t.} \quad & Ax \quad b, && && (1b)\\
& x_i \ 2 \ f0, 1g, && 8i \ 2 \ B, && (1c)\\
& x_i \ 2 \ \mathsf{Z}, && 8i \ 2 \ I \ n \ B, && (1d)\\
& x_i \ 2 \ \mathsf{R}, && 8i \ 2 \ R \ n \ I, && (1e)
\end{aligned}$$

with B containing the indices of binary variables, I containing indices of integer variables, R containing all variable indices, matrix $A \ 2 \ \mathsf{R}^{m \ \ r}$, vectors $b \ 2 \ \mathsf{R}^m$ and $c \ 2 \ \mathsf{R}^r$. Furthermore, Q is the index set of the constraints.

The natural dates formulation of the single machine scheduling problem uses completion time variables C_j and binary ordering variables y_{jk}. This model looks as following:

$$\begin{aligned}
\min \quad & \sum_{j=1}^{n} w_j C_j && && (2a)\\
\text{s.t.} \quad & C_j + p_k \quad C_k + T(1 \quad y_{jk}), && 8j, k \ 2 \ J : j \ 6= \ k, && (2b)\\
& C_k + p_j \quad C_j + T y_{jk}, && 8j, k \ 2 \ J : j \ 6= \ k, && (2c)\\
& C_j \quad p_j, && 8j \ 2 \ J, && (2d)\\
& y_{jk} \ 2 \ f0, 1g, && 8j, k \ 2 \ J : j \ 6= \ k, && (2e)\\
& C_j \ 2 \ \mathsf{R} \ _0, && 8j \ 2 \ J, && (2f)
\end{aligned}$$

with $T \quad \sum_{j=1}^{n} p_j$.

In theory, this is a complete model, but the modelling theory does not always match practice. Within this model, some variations are possible. Before an algorithm can be given, these variations need to be known, such that an algorithm can adapt to these varieties. Many (subtle) diversities are possible, but only the important ones are discussed here.

First, it is possible to model a scheduling problem with only constraints of type (2b) or (2c). Using only constraints of type (2c), the constraints relating the completion times of jobs j and k looks as following:

$$\begin{array}{ll} C_j + p_k & C_k + T y_{kj} \\ C_k + p_j & C_j + T y_{jk}. \end{array}$$

To guarantee that jobs j and k do not overlap, it is necessary that at least one of the two constraints becomes a tight constraint, meaning that the big-M part of the constraint must be set to zero. Therefore, it is needed that the binary variables in the constraints are related through a complementarity constraint:

$$y_{jk} + y_{kj} = 1. \tag{6}$$

Furthermore, it may be that some precedence constraints are present in the scheduling problem, meaning that job j is forced to be scheduled before job k. In a constraint, that looks as follows:

$$C_k \quad C_j \quad p_k. \tag{7}$$

Now that the possible variants of constraints are known, it is time for the next step. As in the input MIP the variables are not labeled with C_j and y_{jk}, but only with x_i, it is helpful to reformulate the constraints in the way that they can be found in an input MIP. Below, a brief overview of how the constraints (2b), (2c), (6) and (7) look in the input is given:

$$\begin{array}{rlll} (2b): & x_i \quad x_j \quad T x_k \quad b_\ell & \text{with } i,j \in R \setminus B, k \in B, b_\ell < 0, \ell \in Q & (8a) \\ (2c): & x_i \quad x_j + T x_k \quad b_\ell & \text{with } i,j \in R \setminus B, k \in B, b_\ell \quad 0, \ell \in Q & (8b) \\ (7): & x_i \quad x_j \quad b_\ell & \text{with } i,j \in R \setminus B, b_\ell \quad 0, \ell \in Q & (8c) \\ (6): & x_i + x_j = 1 & \text{with } i,j \in B & (8d) \end{array}$$

Note that the constraints might be multiplied with a constant $c \in \mathbb{R} \setminus \{0\}$.

Based on these sets, some useful notation for the next section is introduced. Every constraint of one of the above types can uniquely be represented by its type and indices. For example, a constraint of type (8a) with binary variables with indices i, j and non-binary variable index k and constraint index ℓ can be represented as $(8a)_{ijk\ell}$.

3.2 Constraint graphs

As the input of the algorithm is known, the focus can be shifted to the algorithm. An important concept that will be used in the algorithm is that of constraint

graphs. Within the scheduling literature, this is not a new phenomenon, as precedence relations in a scheduling problem have a natural transformation into directed graphs. However, for the goal of the algorithm, a graph for only the precedence relations is not enough. Hence, the concept of a constraint graph for regular big-M constraints is also presented here.

First, the constraint graph based on precedence relations is defined. This *Precedence Digraph* D_p has all non-binary variables indices as nodes and has exactly those arcs (i, j) for which for some ℓ a constraint $(8c)_{ji\ell}$ is present in the input MIP. Every arc in the graph is labeled with two things: as weight the arc gets the value of p_j in the constraint and the second label is the index of the constraint. Summarized in a definition, this looks as follows:

Definition 3.1 (Precedence Digraph)**.** The *Precedence Digraph* $D_p = (V_p, A_p)$ is defined by the following sets:

$$V_p := R \cap B \tag{9a}$$

$$A_p := \{(i, j) : \text{constraint } (8c)_{ji\ell} \text{ is present in input MIP}\} \tag{9b}$$

Every arc (i, j) has the following attributes:

$$w_{ij} := b_\ell \qquad\qquad \ell_{ij} := \ell.$$

The second constraint graph is the constraint graph for the regular (non-precedence) relations between pairs of jobs. This *Regular Digraph* D_r has all non-binary variable indices as nodes and has exactly those arcs (i, j) for which for some k, ℓ a constraint $(8a)_{jik\ell}$ or $(8b)_{jik\ell}$ is present in the input MIP. Also in this graph, every arc is labeled with the value of the processing time and the constraint index. As a third label, every arc gets the index of the binary variable that is present in the constraint. Formally, that looks like this:

Definition 3.2 (Regular Digraph)**.** The *Regular Digraph* $D_r = (V_r, A_r)$ is defined by the following sets:

$$V_r := R \cap B \tag{10a}$$

$$A_r := A^{(0)} \cup A^{(1)} \tag{10b}$$

$$A^{(0)} := \{(i, j)k, \ell : \text{constraint } (8a)_{jik\ell} \text{ is present in input MIP}\} \tag{10c}$$

$$A^{(1)} := \{(i, j)k, \ell : \text{constraint } (8b)_{jik\ell} \text{ is present in input MIP}\} \tag{10d}$$

The attributes of every arc (i, j) are the following:

$$w_{ij} := \begin{cases} T + b_\ell & \text{if } (i,j) \in A^{(0)} \\ b_\ell & \text{if } (i,j) \in A^{(1)} \end{cases} \qquad y_{ij} := k \qquad \ell_{ij} := \ell.$$

Note in both definitions that the processing time p_j, as label referred to as w_{ij}, is not clearly present in (8a)-(8c). However, p_j can be calculated by relating b_ℓ in those equations to their counterparts: the right-hand sides of (2b), (2c) and (7).

Example 3.1. Suppose that a modeller has created a MIP that contains the following constraints:

$$x_1 \quad x_2 \quad 13x_3 \quad 10 \tag{11a}$$
$$x_2 \quad x_4 + 13x_5 \quad 4 \tag{11b}$$
$$x_4 \quad x_1 + 13x_6 \quad 6. \tag{11c}$$

Furthermore, suppose that $RnB = f1, 2, 4g$ and $B = f3, 5, 6g$. These constraints then result in the *Regular Digraph* as given in Figure 1. Constraint (11a) is the

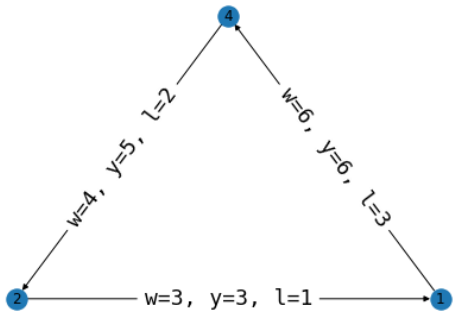


Figure 1: *Regular Digraph* resulting from constraints (11a)-(11c)

constraint that resulted in the arc $(2, 1)$. As $1, 2\ 2\ RnB, 3\ 2\ B$, the coefficients of the non-binary variables are each other's additive inverse and the coefficient of the binary variable and the right-hand side are negative, constraint (11a) can be represented as $(8a)_{1,2,3,1}$, meaning that $(2, 1)\ 2\ A^{(0)}$. From here, the attributes of $(2, 1)$ follow easily: as $(2, 1)\ 2\ A^{(0)}, w_{2,1} = T + b_\ell = 13 \quad 10 = 3$. Furthermore, $y_{2,1} = k = 3$ and $\ell_{2,1} = \ell = 1$. In similar fashion, one can find that the arcs $(1, 4)$ and $(4, 2)$ are part of this *Regular Digraph* and the attributes of those arcs.

3.3 An algorithm to recognize single machine scheduling subproblems

Now that the constraint graphs are defined, the following step is to look at what the minimal output of the algorithm should be. As more than one SMSSP might be present in any MIP instance, the following list of output requirements applies to every returned SMSSP:

1. The set J of nodes that together form a SMSSP. Notice that $J \quad R$.
2. The map $p : J \ ! \ \mathsf{R}_{\ 0}$. In this map the processing times are saved: the x variable with index j has as processing time the value p_j.

Algorithm 1 Find SMSSPs in MIPs (general)

1: **input** MIP M
2: Compute the constraint graphs D_r and D_p from M.
3: Compute transitive closure of the fixed constraint graph D_p to obtain all implied precedence relations.
4: Check for every pair of nodes in the regular constraint graph D_r whether the binary variables in the constraints that the edges represent are properly related and if that is not the case, remove the arcs. The binary variables of constraints of the same type ($((i,j),(j,i)$ 2 $A^{(0)}$ or $(i,j),(j,i)$ 2 $A^{(1)}$) are properly related if they are forced to take complementary values. The binary variables of constraints of different types are properly related if they are forced to take equal value. Also remove arcs if there are no arcs present in both directions.
5: Merge the modified constraint graphs D_r and D_p, turn it into an undirected graph and find all maximal cliques in this graph. Set $L = ;$.
6: For every maximal clique of size at least 2, compute (J, p, g), where J is the set of nodes, p the map for the processing times and g the map for the binary variables. Add (J, p, g) to L.
7: **return** L

3. A map g : f(i,j)ji, j 2 J, i 6= jg ! B [f , , =, :g. Through this map one can find which x variables are the binary variables that relate completion time variables: the x variable with the index b 2 B that has the argument (i, j) models whether job i is scheduled before job j. However, some binaries might not be present. Therefore, the additional objects are used to store which type(s) of constraint(s) are present that do not need a binary variable. If (i, j) is mapped to , job i is required to be scheduled before job j. If (i, j) is mapped to , job i is forced to be scheduled after job j. If (i, j) is mapped to =, job i and j are forced to take equal value. If (i, j) is mapped to :, the variable that models whether job i is scheduled before job j is not present, but its complement, the variable that models whether job j is scheduled before job i, is present.

In some cases of scheduling problems, also release and due dates are present. When that occurs, the algorithm should return this information. This data could turn out to be helpful, as it poses extra restrictions on the SMSSP, which could be exploited to solve the MIP model faster. For the sake of presenting and explaining the algorithm in the simplest way possible, this output requirement is dropped.

In this section, the algorithm is presented twice. Algorithm 1 introduces a high-level description of the algorithm, while Algorithm 2 displays the algorithm in pseudocode. The idea behind the algorithm is to represent the SMSSP as a graph. The jobs form the set of nodes, while a scheduling relation between any pair of jobs, guaranteeing that the jobs can not overlap, is represented by an arc. Then, any clique in such a graph represents a SMSSP. As any submaximal clique

Algorithm 2 Find SMSSPs in MIPs (detailed)

Input Some MIP M
Compute D_r and D_p from M.
Compute transitive closure of D_p; for computed arcs: $\ell_{ij} =$ *implied* and $w_{ij} = \max_{\text{path } (x_i,\ldots,x_j)} w_{ik} + \quad + w_{lj}$.
for $(i,j) \in A^{(s)}, (j,i) \in A^{(t)}$ for $s,t \in \{0,1\}$ **do**
  if $s = t, @\ (8\text{d})_{y_{ij},y_{ji}}$ **then**
    $A_r \quad A_r \setminus \{(i,j),(j,i)\}$
  end if
  if $s \neq t, y_{ij} \neq y_{ji}$ **then**
    $A_r \quad A_r \setminus \{(i,j),(j,i)\}$
  end if
end for
for $(i,j) \in A_r, (j,i) \notin A_r$ **do**
  $A_r \quad A_r \setminus \{(i,j)\}$
end for
$D_c \quad D_r \cup D_p$
Find list C of all maximal cliques in undirected version of D_c.
procedure FindInformation(J)
  for $j \in J$ **do**
    if $\{w_{ij} : (i,j) \in D_c, i \in J\} \neq \ ;$ **then**
      $p_j \quad \min_{(i,j) \in D_c, i \in J} w_{ij}$
    else
      $p_j \quad \max\{0, \text{lower bound of } x_j\}$
    end if
    for $i \in J, i \neq j$ **do**

$$g_{ji} \quad \begin{cases} & \text{if } (j,i) \in A_p, (i,j) \notin A_p \\ & \text{if } (i,j) \in A_p, (j,i) \notin A_p \\ = & \text{if } (i,j),(j,i) \in A_p \\ y_{ji} & \text{if } (j,i) \in A^{(0)}, (i,j),(j,i) \notin A_p \\ y_{ij} & \text{if } (i,j) \in A^{(1)}, (i,j),(j,i) \notin A_p \\ : & \text{otherwise} \end{cases}$$

    end for
  end for
  return (J,p,g)
end procedure
$L \quad ;$
for each set of nodes $J \in C$ with $|J| \quad 2$ **do**
  $(J,p,g) \quad$ FindInformation(J)
  $L \quad L \cup (J,p,g)$
end for
return L

forms a subset of the set of jobs of a scheduling problem, the only interesting cliques are maximal cliques. For the proof that the algorithm indeed finds SMSSPs, see Section 3.4.

In order to show how the algorithm functions, the following example walks through the algorithm step by step.

Example 3.2. Suppose that the following constraints are part of some MIP:

$$x_1 \quad x_2 \quad 15x_3 \quad 12 \tag{12a}$$
$$x_2 \quad x_1 \quad 15x_4 \quad 11 \tag{12b}$$
$$x_3 + x_4 = 1 \tag{12c}$$
$$x_5 \quad x_2 \quad 15x_6 \quad 9 \tag{12d}$$
$$x_2 \quad x_5 + 15x_6 \quad 5 \tag{12e}$$
$$x_5 \quad x_1 + 15x_7 \quad 6 \tag{12f}$$
$$x_1 \quad x_5 + 15x_8 \quad 3 \tag{12g}$$
$$x_2 \quad x_9 + 15x_{10} \quad 4 \tag{12h}$$
$$x_9 \quad x_2 + 15x_{11} \quad 2 \tag{12i}$$
$$x_{10} + x_{11} = 1 \tag{12j}$$
$$x_9 \quad x_5 \quad 2 \tag{12k}$$
$$x_1 \quad x_9 \quad 3 \tag{12l}$$
$$x_{12} \quad x_5 \quad 6 \tag{12m}$$
$$x_1 \quad x_{12} \quad 3 \tag{12n}$$
$$x_2 \quad x_{13} \quad 15x_{14} \quad 11 \tag{12o}$$
$$x_{13} \quad x_2 + 15x_{15} \quad 7 \tag{12p}$$
$$x_1 \quad x_{13} + 15x_{16} \quad 3. \tag{12q}$$

Suppose additionally that $R \setminus B = \{1, 2, 5, 9, 12, 13\}$ and $B = \{3, 4, 6, 7, 8, 10, 11, 14, 15, 16\}$.

Then, the first step of the algorithm is to compute the constraint graphs D_r and D_p. As Example 3.1 shows how constraint graphs can be created from MIP constraints, the calculation of these graphs is not given here. The resulting graphs D_p and D_r are given in Figures 2 and 3. Furthermore, the labels of the arcs in D_r are given in Table 2.

In the next phase, the algorithm computes the transitive closure of D_p. The transitive closure of a graph is the graph that has an arc from node i to node j if node j can be reached from node i in the original graph. Considering Figure 2, one can see that all nodes that can be reached from nodes 1, 9 and 12 are already connected with an arc to those nodes. However, from node 5 all other nodes can be reached, while there is not an arc $(5, 1)$ present in D_p. Hence, a new arc $(5, 1)$ is added while computing the transitive closure. According to line 3 of Algorithm 2, $w_{5,1} = \max_{\text{path } (x_5, \ldots, x_1)} w_{5,k} + \quad + w_{l,1} = \max\{6 + 3, 2 + 3\} = 9$ and $\ell_{5,1} = \textit{implied}$. The resulting transitive closed graph is given in Figure 4.

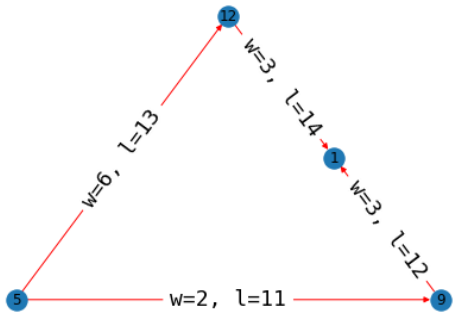


Figure 2: *Precedence Digraph* D_p resulting from constraints (12a)-(12q), leaving out the nodes without arcs 2 and 13

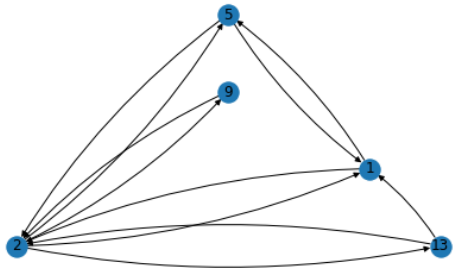


Figure 3: *Regular Digraph* D_r resulting from constraints (12a)-(12q), leaving out the node without arcs 12. For labels corresponding to the arcs, see Table 2

In the subsequent part of the algorithm, the digraph D_r is checked. For every pair of nodes, the algorithm checks whether the binary variables, that are associated with the arcs between a pair of nodes, are forced to take equal value, are constrained to take complementary value or are not related at all. Depending on the type of constraints the arcs represent ((8a) and/or (8b)), the arcs are preserved or removed from the graph.

The first pair of nodes checked in this example is the pair $f1, 2g$. From Table 2 it can be seen that the binary variables associated with the constraints are the variables x_3 and x_4. These variables are related through constraint (12c), from which it follows that the variables are required to take complementary value. As indicated in Table 2, the constraints implying the arcs $(1, 2)$ and $(2, 1)$ have the same type. Since the constraints are in the same class and the binary variables must take complementary values, the arcs are not removed from the digraph. Indeed, when examining constraints (12a)-(12c), it can be seen as follows that the jobs can not overlap:

Table 2: Labels of arcs in D_r (see Figure 3).

Constraint	Type	Arc (i,j)	w_{ij}	y_{ij}	l_{ij}
(12a)	(8a)	(2,1)	3	3	1
(12b)	(8a)	(1,2)	4	4	2
(12d)	(8a)	(2,5)	6	6	4
(12e)	(8b)	(5,2)	5	6	5
(12f)	(8b)	(1,5)	6	7	6
(12g)	(8b)	(5,1)	3	8	7
(12h)	(8b)	(9,2)	4	10	8
(12i)	(8b)	(2,9)	2	11	9
(12o)	(8a)	(13,2)	4	14	15
(12p)	(8b)	(2,13)	7	15	16
(12q)	(8b)	(13,1)	3	16	17

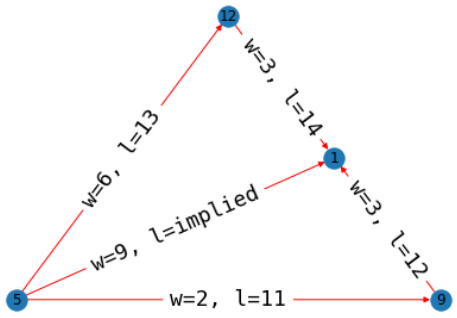


Figure 4: Transitive closure of D_p, leaving out the nodes without arcs 2 and 13

Case 1: Suppose x_3 takes value 0. By (12c), x_4 takes value 1 and constraints (12a),(12b) reduce to

$$\begin{array}{llr} x_1 & x_2 & 12 \\ x_2 & x_1 & 4. \end{array}$$

This means that the completion time of job 2 must be at least 4 larger than the completion time of job 1. Consequently, the jobs can not overlap, assuming that the processing time of job 2 is not larger than 4.
Case 2: Suppose x_3 takes value 1. By (12c), x_4 takes value 0 and constraints (12a),(12b) reduce to

$$\begin{array}{llr} x_1 & x_2 & 3 \\ x_2 & x_1 & 11. \end{array}$$

This means that the completion time of job 1 must be at least 3 larger than

the completion time of job 2. Consequently, the jobs can not overlap, assuming that the processing time of job 1 is not larger than 3.

From these two cases it then follows that the jobs 1 and 2 can not overlap, no matter in which order the jobs are processed. Hence, the presence of the arcs is justified.

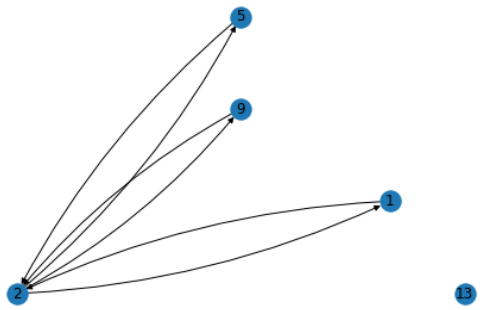


Figure 5: *Regular Digraph* D_r after the arcs are checked, leaving out the node without arcs 12

In similar fashion all the other pairs of nodes can be checked as well. It goes too far to walk through all pairs, but two cases are examined in more detail here.

When studying the pair $\{1, 5\}$, one finds from Table 2 that the constraints corresponding to the arcs are in the same class, while the binary variables x_7 and x_8 are not connected through any other constraints. This means that these corresponding constraints do not guarantee that jobs 1 and 5 do not overlap. For example, both natural date variables can take equal value, while both binary variables take value 1:

$$\begin{array}{rrr} x_5 & x_1 + 15x_7 & 6 \\ & 15 & 6 \end{array} \qquad\qquad \begin{array}{rrr} x_1 & x_5 + 15x_8 & 3 \\ & 15 & 3. \end{array}$$

This validates that the algorithm removes the arcs between nodes 1 and 5.

As can be seen in Figure 3, the node pair $\{1, 13\}$ induces only one arc. As the binary variable x_{16} in the corresponding constraint is not limited in any other constraint, it can take any value. Therefore, this single constraint does not secure that the jobs 1 and 13 do not overlap and so the arc is eliminated from the digraph.

After the algorithm has also checked all other pairs of nodes, the resulting directed graph D_r looks like the graph in Figure 5.

In the following phase of the algorithm, the graphs D_r and D_p are merged into graph D_c. From the undirected version of this graph, as shown in Figure 6, it follows that there are three maximal cliques. The first set of nodes of a maximal clique is $\{1, 2, 5, 9\}$, the second set is $\{1, 5, 12\}$ and the third set is

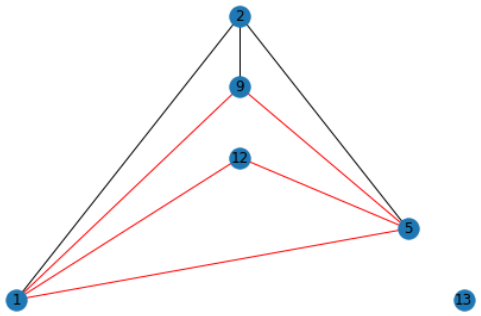


Figure 6: Undirected version of merged graph D_c. Arcs originating from D_r are colored black, while arcs established from D_p are colored red

$\{13\}$. Then the algorithm is concluded with computing the maps p and g for the first two maximal cliques only, as the last maximal clique has size 1.

Table 3: Values of the map p for maximal clique $\{1, 2, 5, 9\}$.

j	p_j
1	3
2	4
5	6
9	2

Table 4: Values of the map g for maximal clique $\{1, 2, 5, 9\}$.

g_{ij} (i \ j)	1	2	5	9
1	-	4		
2	3	-	6	10
5		⋮	-	
9		11		-

Here, the computation of some values of p and g is given for maximal clique $\{1, 2, 5, 9\}$. The first value that is computed here, is the value for p_2. Following line 20 of Algorithm 2, it must be that $p_2 = \min_{(i,2) \in D_c} w_{i2} = \min\{w_{1,2}, w_{5,2}, w_{9,2}\}$. Filling in the values of the weights of the arcs, leads to $p_2 = \min\{4, 5, 4\} = 4$. The fact that the processing time differs between constraints, could for example be explained by sequence-dependent setup times.

The next value shown here, is that of p_1. In similar fashion as above, it has to be that $p_1 = \min_{x_{i6=1} \in J} w_{i1} = \min\{w_{2,1}, w_{5,1}, w_{9,1}\} = \min\{3, 9, 3\} = 3$. Here, the different possible processing times are caused by the implied precedence relation $5 \; ! \; 1$.

Then, also some values of g are calculated here. The computations follow line 25 of algorithm 2. The first examples are $g_{1,5}$ and $g_{1,5}$. As can be seen from Figure 4, there is a precedence relation $5 \; ! \; 1$ present between those jobs, meaning that $(5, 1) \in A_p$. This then means that $g_{1,5} =$ and $g_{5,1} =$.

The values of $g_{2,5}$ and $g_{5,2}$ follow again line 25 of algorithm 2. From Figure

4 it becomes clear that there is no precedence relation between this pair of jobs, so from Table 2 it can be seen that arc $(2,5)$ 2 $A^{(0)}$, as it follows from a type (8a) constraint, while $(5,2)$ 2 $A^{(1)}$. Therefore, $g_{2,5} = 6$ and $g_{5,2} = :$.

The full set of values of p and g for maximal clique $f1,2,5,9g$ can be found in Tables 3 and 4. Note that g is not defined for g_{ij} with $i = j$.

Table 5: Values of the map p for maximal clique $f1,5,12g$.

j	p_j
1	3
5	0
12	6

Table 6: Values of the map g for maximal clique $f1,5,12g$.

g_{ij} (i \ j)	1	5	12
1	-		
5		-	
12			-

In the other relevant maximal clique $f1,5,12g$ an unique situation arises. This clique contains only precedence relations and no regular scheduling relations. As there is a job that precedes all other jobs, in this case job 5, there are no arcs $(i,5)$ 2 D_c $8i$ 2 J. Then, it is not possible to compute p_5 according to line 20 of Algorithm 2 and line 22 of Algorithm 2 is the alternative. Since variable x_5 is assumed to be a completion time variable, it should have a lower bound of at least the processing time of the job plus possibly a release date. As job 5 is forced to be processed first, it does not matter whether the lower bound consists of a release date and a processing time or only a processing time and therefore the lower bound serves as a good source to find the processing time. As in this example the lower bounds are not explicitly given and for the real variables are assumed to be 1, the algorithm takes 0 as processing time. For the full set of values of p and g for this maximal clique, see Tables 5 and 6.

3.4 Validation of the recognition algorithm

In this section, it is shown that the algorithm as presented in Section 3.3 indeed finds SMSSPs that are part of the input MIP. This is formalized in the following theorem:

Theorem 3.1. *For each tuple* (J,p,g) *found by* Algorithm 2*, the set of* x *variables with indices* j 2 J *form a set of completion time variables that are part of a SMSSP of the input MIP.*

Proof. To be able to prove this theorem, assume the algorithm has computed some instance (J,p,g) 2 L. Furthermore, assume the MIP takes a feasible solution such that at least one pair of variables fx_i, x_jg in the input MIP with i,j 2 J takes values such that the corresponding jobs overlap. Finally, assume w.l.o.g. that job j is the job that is processed last. As the jobs overlap, this means that $p_j > 0$. Then, one can summarize the assumptions with the following strict inequalities: $x_j \quad x_i < p_j$ and $x_j > x_i$. As Algorithm 2 in line 20 computes

$p_j = \min_{x_k \in J} w_{kj}$, it follows that $x_j - x_i < \min_{x_k \in J} w_{kj} - w_{ij}$. Then, there must be an arc (i,j) and/or (j,i) in (one of) the underlying graphs.
Case 1: $(i,j) \in A_p, \ell_{ij} \neq$ *implied*.

$$\begin{aligned} & x_j - x_i < w_{ij} \\ & x_j - x_i < b_\ell, && \text{by definition of the label } w_{ij}, \\ & x_j - x_i < x_i - x_j, && \text{by (8c)}, \\ & x_j < x_i, \end{aligned}$$

which is a contradiction to the assumption $x_j > x_i$.
Case 2: $(i,j) \in A_p, \ell_{ij} =$ *implied*.

$$\begin{aligned} & x_j - x_i < w_{ij} \\ & x_j - x_i < w_{ik} + \cdots + w_{\ell j}, && \text{by step 3 of Algorithm 2}, \\ & x_j - x_i < b_{\ell_1} + \cdots + b_{\ell_p}, && \text{by definition of the label } w_{ij}, \\ & x_j - x_i < (x_i - x_k) + \cdots + (x_\ell - x_j), && \text{by (8c)}, \\ & x_j - x_i < x_i - x_j, \\ & x_j < x_i, \end{aligned}$$

which is a contradiction to the assumption $x_j > x_i$.
Case 3a: $(i,j) \in A^{(0)}, (j,i) \in A^{(1)}, x_k = 1$.

$$\begin{aligned} & x_j - x_i < w_{ij} \\ & x_j - x_i < T + b_\ell, && \text{by definition of the label } w_{ij}, \\ & x_j - x_i < x_i - x_j, && \text{by (8a)}, \\ & x_j < x_i, \end{aligned}$$

which is a contradiction to the assumption $x_j > x_i$.
Case 3b: $(i,j) \in A^{(0)}, (j,i) \in A^{(1)}, x_k = 0$.
For arc (j,i), implying constraint $(8b)_{ijk\ell}$, the inequality (8b) reduces to

$$x_i - x_j \geq b_\ell \qquad \text{for some } b_\ell > 0.$$

This inequality thus means that $x_i - x_j$, which is a contradiction to the assumption $x_j > x_i$.

The reasoning in cases 3a and 3b can be applied to the six other cases where (i,j) and (j,i) are in $A^{(0)}$ or $A^{(1)}$ and binary x_k takes value 0 or 1. This means that all the eight cases lead to a contradiction. That implies that the assumption, that the MIP takes a feasible solution such that at least one pair of variables $\{x_i, x_j\}$ takes values such that the corresponding jobs overlap, leads to a contradiction and therefore the assumption can not hold true. This means, that for any feasible solution to the MIP the set of x variables with indices $j \in J$ form a set of completion time variables that are part of a SMSSP of the input MIP. □

3.5 Complexity of the recognition algorithm

To be able to later say something useful about the performance of the algorithm, it is good to look at the theoretical complexity of the algorithm as defined in Section 3.3. Therefore, in this section it is shown that the following Proposition holds true.

Proposition 3.1. *The complexity of* Algorithm 2 *is* $O(m + r^2 \ 3^{r/3})$*, with* m *the number of constraints and* r *the number of variables in the input MIP.*

Proof. The first part of the algorithm is to compute the *Regular* and *Precedence Digraphs* D_r and D_p. Therefore, it is necessary to loop over all the constraints of a MIP, such that it can be checked for every constraint whether it implies an arc in one of the constraint graphs. This then can be done in $O(m)$ time, as any constraint that has more than 3 variables can be discarded as constraint that implies an arc.

The next step is to calculate the transitive closure of D_p, including a calculation of longest path lengths. As by negating the weights of all the edges the longest path problem can be converted to a shortest path problem, the Floyd-Warshall algorithm can be used to compute the transitive closure and the corresponding data in $O(r^3)$ time [22].

In the following phase, the binaries corresponding to the arcs in D_r are checked. In a straightforward implementation, where a loop over all pairs of nodes of D_r contains a loop over all constraints of the MIP, this can be done in $O(r^2 m)$ time. However, as in the first part of the algorithm all constraints are already looped over, all constraints of type (8d) can be found already then and stored in the memory. Therefore a loop over all constraints of the MIP can be replaced by a simple check, decreasing complexity to $O(r^2)$.

The succeeding stage of the algorithm is to find all maximal cliques in the undirected version of the merged graphs D_p and D_r. The complexity for an algorithm to solve this problem is $O(3^{r/3})$ [23]. This complexity is optimal, as there exist at most $3^{r/3}$ maximal cliques in a graph with r vertices.

In the final part of the algorithm, the maps p and g for the processing times and binary variables are computed. Per clique, this takes $O(r^2)$ time for computing map p, but also for computing map g. As there exist up to $3^{r/3}$ maximal cliques in a graph, the total complexity of the final part is $O(r^2 \ 3^{r/3})$.

Then, the complexity of the full algorithm is $O(m+r^3+r^2+3^{r/3}+r^2 \ 3^{r/3}) = O(m + r^2 \ 3^{r/3})$. □

4 Scheduling cuts

As there now is the possibility to recognize a SMSSP in a MIP, the next step is to exploit this substructure. Exploiting a substructure in a MIP means to generate and add cuts based on the substructure. Therefore, in this section some scheduling cuts that can be added to a MIP are introduced.

In scheduling literature, many sets of valid inequalities for the scheduling polytope can be found. However, not all of these inequalities are suitable as cutting plane, so some selection criteria are helpful to decide which inequalities should be made use of:

1. An inequality should contain at least one variable that is present in the MIP.
2. An inequality should not introduce any new variables.
3. The number of inequalities within one set of inequalities should grow at most polynomial in terms of the number of jobs in a scheduling problem.

The first criterion is a minimal requirement for an inequality to possibly cut off some feasible solutions. Without the second demand, the original set of feasible solutions of a MIP needs modification and the number of feasible solutions most likely increases instead of decreases by adding such an inequality. This then slows the solving of MIPs down instead of speeding it up. The final criterion is not necessary, but is a matter of choice. This choice is made to first see whether cutting planes that are guaranteed to be generated in polynomial time already decrease solving time significantly. In a later stage, when it turns out that the used cutting planes do not decrease solving time largely, this last criterion can be dropped.

The cutting planes that remained after selection can be sorted in two categories: cuts that are based on release dates and cuts that are based on precedence relations.

4.1 Release date cuts

In this section, it is assumed that the jobs are ordered by ascending release date, i.e. $r_1 \quad r_2 \qquad r_n$.

An important source of valid inequalities for single machine scheduling problems using natural date variables is the paper by Nemhauser and Savelsbergh [14]. Most of the valid inequalities presented in this paper are subset-based inequalities, so the number of inequalities grows exponential in the number of jobs in the scheduling problem and therefore these inequalities did not meet the selection criteria. However, two sets of inequalities remain and are discussed here.

The first set of inequalities is the set of dominance cuts. These inequalities are defined as follows:

Definition 4.1 (Dominance inequalities)**.** The dominance inequalities are defined by the following inequality:

$$\sum_{i \in J: r_i \leq \alpha_j} y_{ij} \geq 1 \tag{16}$$

with $\alpha_j := \min_{k \in J: r_k + p_k \leq r_j} (r_k + p_k)$.

Note here that α_j takes the same value for all jobs j, but that α_j is not defined for all jobs j. This means that also the inequality itself is not defined for all jobs.

Proposition 4.1. *Inequality* (16) *is valid for the single machine scheduling problem using natural date variables.*

Proof. From Definition 4.1 it is known that $\alpha_j := \min_{k \in J: r_k + p_k \leq r_j} (r_k + p_k)$. This means that α_j equals the earliest possible completion time of any job $k \in j$, for jobs j that have a release date later than this earliest possible completion time. As then job j is released later than this earliest possible completion time, one is sure that in an optimal schedule at least one job i with a release date before the earliest possible completion time is scheduled before job j. From applying this reasoning to the model with natural date variables, inequality (16) follows. □

Furthermore, Nemhauser and Savelsbergh [14] presented a linear programming formulation, showing great similarities with a set of valid inequalities presented by Dyer and Wolsey [12], which will not be given here. The linear programming formulation by Nemhauser and Savelsbergh consists of a set of lower bounds on starting time variables S_j. However, as the natural dates formulation (2) is defined with completion time variables, S_j is replaced with $C_j - p_j$. Then, a set of release date inequalities can be defined.

Definition 4.2 (Release date inequalities)**.** The release date inequalities are defined by the following set of equations:

$$\begin{aligned} C_j \geq\ & p_j + r_i + \sum_{k<i, k \neq j, r_k + p_k \leq r_i} p_k (y_{ik} + y_{kj} - 1) \\ & + \sum_{k<i, k \neq j, r_k + p_k > r_i} [(r_i - r_k)(y_{ik} + y_{kj} - 1) + (r_k + p_k - r_i) y_{kj}] \\ & + \sum_{k \geq i, k \neq j} p_k y_{kj}, \qquad 1 \leq i \leq j \leq n, \end{aligned} \tag{17a}$$

$$\begin{aligned} C_j \geq\ & p_j + r_j + (r_i - r_j) y_{ij} + \sum_{k<i, k \neq j, r_k + p_k \leq r_i} p_k (y_{ik} + y_{kj} - 1) \\ & + \sum_{k<i, k \neq j, r_k + p_k > r_i} [(r_i - r_k)(y_{ik} + y_{kj} - 1) + (r_k + p_k - r_i) y_{kj}] \\ & + \sum_{k \geq i, k \neq j} p_k y_{kj}, \qquad 1 \leq j < i \leq n. \end{aligned} \tag{17b}$$

At first glance, these inequalities seem complicated. However, both these inequalities are strengthened versions of the same set of inequalities:

$$C_j \quad p_j \quad r_i y_{ij} + \sum_{k<i,k\neq j} p_k(y_{ik} + y_{kj} \quad 1) + \sum_{k \quad i,k\neq j} p_k y_{kj}, \quad 1 \quad i,j \quad n. \quad (18)$$

Proposition 4.2. *Inequalities* (18) *are valid for the single machine scheduling problem using natural date variables.*

Proof. Every inequality of the form of inequalities (18) forms a lower bound on the starting time of job j and is related to job i. The lower bound consists of three parts. Per part it is shown that it is correct to add this to the lower bound, which is sufficient to show the validity of the inequalities.

The first term of the right-hand side adds the release date of job i to the lower bound, but only when job i is scheduled before job j. This last addition validates the presence of this term in the lower bound.

The first sum concerns those jobs k that have a release date that is at most equal to the release date of job i. For those jobs, the processing time p_k is added to the lower bound only if job k is processed after job i and before job j. It is necessary to check whether job k is processed after job i, as job k might be processed before job i is released. Therefore, the addition of this sum to the lower bound is justified.

The last term of the right-hand side finally adds the processing time of all jobs k with a release date at least equal to the release date of job i. However, p_k is only added when job k is processed before j, which confirms the legitimacy of this part of the lower bound. □

Note that the bound can become a loose bound in two ways. In the first sum it happens that p_k is added to the sum when job k is processed after job j and before job i due to the linearization of $y_{ik}y_{kj}$ to $(y_{ik} + y_{kj} \quad 1)$. Similarly, the first part is reduced to 0 when job j is processed before job i. However, in this case a loose bound is also a valid bound and thus does not cause any real trouble.

Corollary 4.2.1. *Inequalities* (17a) *and* (17b) *are valid for the single machine scheduling problem using natural date variables.*

Proof. Here, it is shown that inequalities (17a) and (17b) are valid strengthened versions of inequalities (18). The corollary thus can be shown to hold true by showing that the improvements are valid.

The first enhancement concerns the first element of the right-hand side of (18): $r_i y_{ij}$. When $i \quad j$, meaning $r_i \quad r_j$, r_i can not only be added to the lower bound if job i is scheduled before job j, but also if job j is scheduled before job i. Thus, when $i \quad j$, $r_i y_{ij}$ can be replaced by r_i. In the case that $i > j$, implying $r_i \quad r_j$, r_i indeed can only be added when job i is scheduled before job j. However, when job j is scheduled before job i, the first element does not have to be 0, it can actually be r_j. Therefore, if $i > j$, $r_i y_{ij}$ can be substituted by $(r_i \quad r_j)y_{ij}$.

The second reinforcement is related to the second term of the right-hand side of (18): $\sum_{k<i,k\neq j} p_k(y_{ik} + y_{kj} \quad 1)$. In the case that $r_k + p_k \quad r_i$, this term is already optimal. In the case that $r_k + p_k > r_i$, this term can be strengthened. In this instance, the part of job k that must be processed after job i is released, exactly $r_k + p_k \quad r_i$, can already be added when it is given that job k is processed before job j without requiring that job k is also processed after job i. On the other hand, the part of job k that could be processed before job i is released, precisely $r_k \quad r_i$, still also requires that job k is processed after job i. So, $\sum_{k<i,k\neq j} p_k(y_{ik} + y_{kj} \quad 1)$ can be replaced by $\sum_{k<i,k\neq j,r_k+p_k \quad r_i} p_k(y_{ik} + y_{kj} \quad 1) + \sum_{k<i,k\neq j,r_k+p_k>r_i} [(r_i \quad r_k)(y_{ik} + y_{kj} \quad 1) + (r_k + p_k \quad r_i)y_{kj}]$. □

Note that the first improvement is defining the difference between inequalities (17a) and (17b), while the second improvement induces an extra term in the right-hand side of the inequality.

4.2 Precedence cuts

In his paper on single machine scheduling problems with precedence relations, Wolsey [24] presents a non-linear inequality for every precedence relation $i \ ! \ j$:

$$C_j \quad C_i \quad p_j + \sum_{k2Jnfi,jg} p_k y_{ik} y_{kj}.$$

In brief, this inequality tells that the difference in completion time of jobs i and j is at least that of the processing time of job j plus the processing time of all jobs k that are processed in between jobs i and j. However, as the inequality is non-linear, Wolsey also came up with a linearized version of this inequality, that leads to the following definition of precedence inequalities.

Definition 4.3 (Precedence inequalities)**.** For every precedence relation $i \ ! \ j$, the precedence inequality is the following:

$$C_j \quad C_i \quad p_j + \sum_{k2S(i)\setminus P(j)} p_k + \sum_{k2S(i)nP(j),k\neq j} p_k y_{kj} + \sum_{k2P(j)nS(i),k\neq i} p_k y_{ik}, \quad (19)$$

with $S(i)$ the index set of jobs succeeding job i and $P(j)$ the index set of jobs preceding job j.

Proposition 4.3. *Inequality* (19) *is valid for the single machine scheduling problem using natural date variables.*

Proof. As job j is scheduled after job i, it is clear that the completion time of job j should be larger than the completion time of job i. This justifies the left-hand side of inequality (19). The right-hand side of the inequality forms a lower bound on the difference between the completion times C_j and C_i and consists of four parts. Per part, it is shown that it is valid to add this to the lower bound, which then implies that the inequality is valid.

First, the processing time of job j is added, which is the minimum amount by which the completion times C_j and C_i should differ and therefore it is logical to add this to the lower bound.

Next, the processing time of all jobs k, of which it is known that is processed after job i (as $k \in S(i)$) and before job j (as $k \in P(j)$), can legitimately be added as well, as job k is processed between jobs i and j.

The third element is to add the processing time of jobs k, of which it is known that it succeeds job i, but of which it is not known whether it precedes job j. However, these processing times are only added when it turns out that job k is indeed processed before job j and hence it is reasonable to add this element to the lower bound.

Finally, the processing times of jobs k, of which it is known that it precedes job j, but of which it is not known whether it succeeds job i, are added, but only when job k indeed succeeds job i. Thus, in similar fashion as the third element, it is correct to add this to the lower bound. □

5 Computational Study

As for the SMSSP the recognition algorithm is defined and some possible cuts are given, the step of exploiting the substructure can be taken. In this section, first the implementation details and results of a test of the recognition algorithm are given. Thereafter, the method and results of generating cuts and solving various (modified) MIP models are provided.

5.1 Recognition algorithm

In this section the implementation details, results and its interpretation are given only for the test of the recognition algorithm as defined in Section 3.3. This includes results on the amount of SMSSPs found by the algorithm, on the size of the subproblems and on the running time of the algorithm. Results on the exploitation of the SMSSPs are given in Section 5.2.

5.1.1 Implementation details

In order to test the recognition algorithm, it is programmed in Spyder 4.2.5 [25] with Python 3.8 [26]. Then, the recognition algorithm is tested on the benchmark set of MIPLIB 2017 [27], consisting of 240 MIP instances. This set of instances is a standard set to test the performance of solving software and contains a representative set of MIP instances. The benchmark set is tested on a single core of an Intel Core i7 CPU running at 2.60 GHz with 16 GB RAM. The time limit for the recognition algorithm to try to find SMSSPs in a single MIP instance was set to 1 hour. To run the graph algorithms that are used in the recognition algorithm, the NetworkX-package [28] is used.

The algorithm is not exactly implemented as it is given in Section 3.3 and as a consequence the theoretical complexity of parts of the algorithm as given in Section 3.5 is also changed. The first difference lies in the definition of the constraint graphs D_p and D_r. According to Definitions 3.1 and 3.2, the full set of indices of non-binary variables forms the set of nodes of the constraint graphs. However, in the implementation only the indices of variables that are involved in constraints that imply arcs in one of the graphs form the set of nodes. Furthermore, according to Definition 3.1, arcs are only implied by inequality constraints. Nevertheless, a modeller might also fix the difference between two completion time variables and therefore use equality constraints. Accordingly, in the implementation, constraints of the following form also imply an arc in precedence graph D_p:

$$x_i \quad x_j = b_\ell \qquad \text{with } i, j \in R \setminus B, b_\ell \quad 0, \ell \in Q.$$

The next difference is found in the transitive closure. In the implementation the Floyd-Warshall algorithm is not used, but instead an all pairs variant of the Bellman-Ford algorithm is executed. Although the theoretical complexity increases from $O(r^3)$ to $O(mr^2)$ [29] with m the number of constraints and r the number of variables in the input MIP, small test runs of both implementations

showed that the implementation of the all pairs version of the Bellman-Ford algorithm performed much better.

Another dissimilarity between the theoretical and practical algorithm concerns the part where the binaries are checked. In the algorithm, only a check for the presence of complementarity constraint containing both binary variables or a check that the binary variables are equal takes place. However, much more complicated binary relations could be present. For example, binary variables might be forced to take equal value by having complementarity constraints with a third binary variable. Furthermore, binary variables could be set to 0 or 1, forcing the variables to take complementary or equal values.

To detect these more complex relations, an undirected auxiliary graph is created in the same part of the algorithm where the constraint graphs D_p and D_r are calculated. The nodes of this auxiliary graph is the set of nodes that represents the binary variables $U = \{u_b | b \in B\}$, together with the set that represent the complementary binary variables $V = \{v_c | c \quad r \in B\}$ and two nodes that represent the fixation to 0 and 1 f_0, f_1. Then, an equality constraint forcing x_i and x_j to take equal value induces edges $\{u_i, u_j\}$ and $\{v_{i+r}, v_{j+r}\}$. A complementarity constraint of type $(8d)_{ij}$ causes edges $\{u_i, v_{j+r}\}$ and $\{v_{i+r}, u_j\}$. A binary variable x_i that is forced to take value 0 leads to edges $\{u_i, f_0\}$ and $\{v_{i+r}, f_1\}$, while binary variable x_j that must take value 1 implies edges $\{u_j, f_1\}$ and $\{v_{j+r}, f_0\}$.

Given this auxiliary graph, the binary check is not a complex process anymore. To check whether two binary variables x_i and x_j are constrained to take complementary values is to check whether there exists a path from node u_i to node v_{j+r} or a path from v_{i+r} to u_j in the auxiliary graph. The check whether two binary variables x_i and x_j are forced to take equal value has reduced to checking whether a path from u_i to u_j exists in the auxiliary graph. As with a simple breadth-first search all connected components of the auxiliary graph can be found, the implementation can be run in $O(m + r)$ time, instead of the theoretical complexity of $O(r^2)$.

The final adjustment in the implementation of the algorithm concerns the assumption of the algorithm that the non-binary variables in the SMSSP are completion time variables. However, it is possible as well that a modeller used starting time variables. Therefore, in the part where maps p and g are computed, in the implementation it is first decided whether the modeller used completion or starting time variables. To understand how this decision can be made, it is good to see how the algorithm processes constraints that use starting time variables instead of completion time variables. The constraint that models a precedence relation using completion time variables is given in inequality (7):

$$C_k \quad C_j \quad p_k.$$

The same type of constraint using starting time variables looks as follows:

$$S_k \quad S_j \quad p_j.$$

As the algorithm considers these constraints with starting time variables to be type $(8c)_{ij}$ constraints, such a constraint will imply an arc (j, k) with weight

$w_{jk} = p_j$ in the *Precedence Digraph*, while inequality (7) implies an arc (j, k) with weight $w_{jk} = p_k$ in the same graph.

A similar argument can be made for the starting time variables variants of the inequalities (2b) and (2c). For all constraints with starting time variables, it holds true that the weight of the associated arc is the processing time that belongs to the variable represented by the tail of the arc, instead of the variable represented by the head of the arc. That means that the formula in Line 20 of Algorithm 2 should be replaced by

$$p_j \qquad \min_{(j,i) \in D_c, i \in J} w_{ji}.$$

Then, the decision whether the model uses completion time or starting time variables can be made by checking the cardinalities of all the sets $\{w_{ji} : (j,i) \in D_c, i \in J\}$ and $\{w_{ij} : (i,j) \in D_c, i \in J\}$. When the inequality below holds true, the model is assumed to use completion time variables, while a violation of this inequality suggests that the model uses starting time variables:

$$\sum_{j \in J} |\{w_{ij} : (i,j) \in D_c, i \in J\}| \quad \sum_{j \in J} |\{w_{ji} : (j,i) \in D_c, i \in J\}|.$$

As the decision for completion or starting time variables can be done in $O(r^2)$ time per maximal clique, this check does not influence the complexity of the final part of the algorithm.

Given the complexity in parts of the algorithm in the implementation, the total complexity of the implementation is $O(m+mr^2+m+r+3^{r/3}+r^2 \ 3^{r/3}) = O(mr^2 + r^2 \ 3^{r/3})$.

5.1.2 Results

Of course, there are many interesting aspects to this experiment, of which the detailed results can be found in Appendix A. The first results shown in Figure 7 are the results on the number of SMSSPs found and the size of the largest subproblem found. In this figure, only the instances in which the algorithm found at least one SMSSP are represented with a dot. In total, the algorithm found SMSSP(s) in 69 out of 240 instances. From Figure 7, it can be seen that the number of SMSSPs found by the algorithm per instance varies from 1 to approximately 60.000. Moreover, the size of the largest SMSSP is below 5 in most of the cases, but in some cases it reaches the value of 40. As for small scheduling problems the cuts most likely do not have any (desired) effect on the solving time of the MIP, it was decided to apply a filter to the SMSSPs that were found. First, a SMSSP should consist of at least three jobs. Furthermore, the maximal processing time or release date of all the jobs in a SMSSP should be larger than 0 for any cut to be effective. The filtered results are given in Figure 8.

The first things that stands out in Figure 8, is the large decrease in amount of instances in which a filtered SMSSP is present: only 9 instances are left over. This means that from Figure 7 most of the instances have disappeared, which

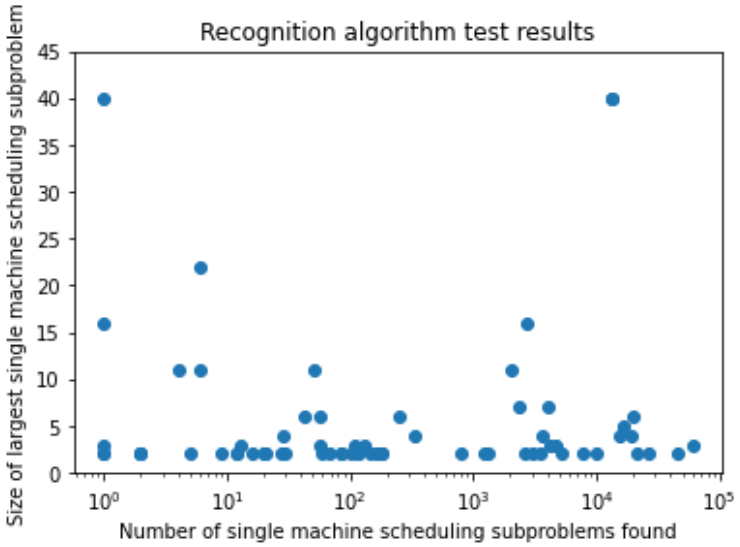


Figure 7: Results of the test of the recognition algorithm on benchmark set of MIPLIB 2017

happens when all of the subproblems are filtered out. Moreover, some of the remaining instances have different results. From Figure 7, some dots have moved to the left and in some cases also down. A move to the left can be explained by removing only some of the SMSSPs, as a part of the subproblems of an instance have made the cut. However, when the SMSSP of the largest size is removed by the filter, the new largest subproblem may have a smaller size, resulting in a dot moving to the left and down. For the instance at $(1, 20)$ in Figure 8 this is the case: it must come from the dot $(6, 22)$ or $(40, 13449)$ in Figure 7. In general, as a result in Figure 8 the number of SMSSPs in one instance is reduced to at most 3500 and the largest SMSSP in some instance now consists of at most 20 jobs instead of 40.

Another relevant aspect of the algorithm, is the amount of time it takes to find SMSSPs. This is important, as the recognition algorithm (and cut generation) should take less time than the time that is gained in solving the MIP. In order to be able to say something useful about the performance of the algorithm, the algorithm is divided in five parts, which were individually timed while running the algorithm:

1. The part where the constraint graphs D_r and D_p are computed. This part is abbreviated as CG. In practice, this part also includes the time Gurobi needs to open and read the model.

2. The part where the transitive closure of *Precedence Digraph* D_p is computed. This part is abbreviated as TC.

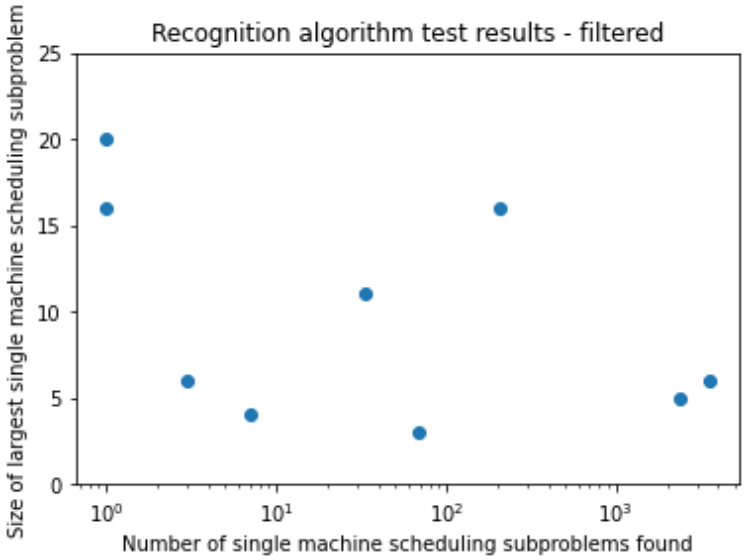


Figure 8: Filtered results of the test of the recognition algorithm on benchmark set of MIPLIB 2017. The filter guarantees that the remaining problems have at least three jobs and at least one processing time or release date larger than 0

3. The part where the binary variables associated with arcs in *Regular Digraph* D_r are checked. This part is abbreviated as CB.

4. The part where constraint graphs are merged and all maximal cliques are computed in the undirected version of the merged graph. This part is abbreviated as FC.

5. The part where maps p and g are computed. This part is abbreviated as CM.

Table 7: Mean and standard deviation of the running time of (parts of) the recognition algorithm for the instances that terminated within one hour.

	Mean running time	Standard deviation of the running time
CG	7.06 10^3 ms	2.46 10^4 ms
TC	37.1 ms	150 ms
CB	2.57 ms	12.0 ms
FC	37.6 ms	154 ms
CM	46.5 ms	238 ms
Total	7.19 10^3 ms	2.47 10^4 ms

During the experiment the algorithm did not meet the time limit of one hour in 3 of 240 cases. For those three cases, the algorithm always passed the hour mark in the FC part. The running time results of the 237 instances for which the algorithm was completed within an hour, are used to calculate the mean and variance of the running time for every part and for the entire algorithm. These numbers are given in Table 7.

From Table 7 it can be seen that the algorithm takes on average a little more than 7 seconds to complete. On average, the CG part of the algorithm takes 7 seconds as well, which makes it highly likely that in most cases the algorithm spends most of its running time in the CG part. The other parts do on average not need more than 0.05 seconds each to complete, with the CB part taking the fewest time on average. Also in terms of standard deviation, the CG part of the algorithm has the largest value, while the CB part takes the lowest value.

5.1.3 Analysis

The first part to analyse is the running time of the algorithm in practice compared to the theoretical complexity. As mentioned in Section 5.1.1, the implementation differs somewhat from the algorithm in theory in the parts where the transitive closure of D_p is computed, where the checks of the binary variables are executed and where the maps p and g are computed. The first difference concerns the computation of the transitive closure of D_p. Theoretically some improvement is possible by executing the Floyd-Warshall algorithm instead of the all pairs variant of the Bellman-Ford algorithm. However, some test runs with both algorithms suggested that the latter algorithm performs better in practice, so it is unlikely that this theoretical improvement leads to better practical performance.

Remark 5.1. *To compute the transitive closure of D_p, the all pairs variant of the Bellman-Ford algorithm likely performs better than the Floyd-Warshall algorithm.*

As opposed to a change in the implementation of the transitive closure of D_p, a simplification of the checks of the binary variables might cause a loss of information. When the implementation matches the description in Section 3.5, more complex relations of the binary variables will not be detected by the algorithm and the corresponding arcs will be removed from D_r. It can easily be checked whether any information is lost by checking the path length as soon as the relation between the binary variables is confirmed. When the path length is 1, this must be the result of one complementarity or equality constraint. Also path length 0 is fine, when the same binary variable is used and the binaries should take equal value. Any longer path lengths are caused by more complex relations. When only path lengths of 0 and 1 are found for the arcs that are not removed from D_r, the simplification of the checks is feasible. However, given the amount of time the algorithm on average spends in the part where the binaries are checked, a big improvement in terms of running time will not be made by simplifying the check. On the other hand, the auxiliary graph is computed in

the CG part and this calculation could give reason to a significant amount of the time spent in this part of the algorithm. To be able to decide whether this check for more complex relations of binary variables is necessary or useful, it is checked whether (a lot of) complex relations are present in the tested models.

Supplementary data show that the auxiliary graphs contained connected components of size up to 60,000 nodes and these large components usually contain the node f_0 or f_1. For the nine models that remained after filtering, the largest connected component in an auxiliary graph consisted of only 5 nodes. Further tests showed the largest connected component that was used to guarantee that an arc was not removed from the *Regular Digraph* was made up of only two nodes. This means that the possible more complicated relations between binary variables as described in Section 5.1.1 were not detected in a SMSSP in practice. Based on this result, it can be argued that the construction of the auxiliary graph and its application to check the binary relations can be removed from the implementation. Its construction in the CG part of the algorithm should be replaced by storing all pairs of binary variables that are complementary and the application in the CB part can be replaced by checking whether a pair of variables is stored.

Remark 5.2. *The use of an auxiliary graph to detect complex binary relations did not a ect the number or size of found SMSSPs and can be removed from the implementation of the recognition algorithm.*

The final difference between the algorithm and the implementation is the check for completion or starting time variables. This check can be included in the algorithm without increasing theoretical complexity, but of course, such a check does add to the running time in practice. Assuming the check is left out and the assumption of completion time variables is made in the algorithm, then a model with starting time variables would most likely have smaller processing times for most of the jobs. Then, exploiting the SMSSP(s) possibly is less helpful, as the generated cutting planes are probably less tight. However, given the amount of time the algorithm spends in the CM part, it is not a check that takes a lot of time in practice and thus is worth executing.

Remark 5.3. *The additional check for completion or starting time variables in the recognition algorithm costs little time, but likely improves the exploitation of SMSSPs.*

A further improvement of the algorithm could be to incorporate the filter, that is applied in Section 5.1.2, in the algorithm itself. This filter influences the algorithm in two ways. First, the filter removes all cliques of size 2. This can be included in the part of the algorithm that finds all maximal cliques. Secondly, the filter removes the SMSSPs that do not have at least one release date or processing time larger than 0. As processing times are taken to be the minimum of weights of the incoming arcs (see Algorithm 2, Line 20), the way to prevent processing times of 0 is to simply not add arcs to the constraint graphs D_p and D_r with a weight of 0 or smaller in the first place.

This approach has a few consequences, the first being that some nodes representing jobs with a relevant release date might be excluded from a SMSSP. Secondly, the inclusion of the filter in the algorithm might actually cause the algorithm to find more relevant, but smaller SMSSPs. This can be seen as follows. There might exist a node with all outgoing arcs with weight 0, which implies that all other nodes/jobs get assigned a processing time of 0. When that node itself also turns out to have a non-positive processing time, the SMSSP would be filtered out. However, when the node with outgoing arcs of weight 0 would not have been included in the SMSSP, the SMSSP may actually turn out to be a useful subproblem.

Another reason to incorporate the filter in the algorithm is the fact that the algorithm as given in Section 3.3 allows arcs induced by type $(8a)_{ij}$ constraints with a negative weight. However, in practice it is of course not possible the have a job with a negative processing time and therefore it stands to reason to not allow such arcs. When the filter would be included, indeed those arcs will be removed.

Additionally, in 3 out of 240 test instances the algorithm needs at least an hour to find all maximal cliques. Of course, with an exponential theoretical complexity for this part of the algorithm, it is no surprise that this happens. However, incorporating the filter decreases the number edges and nodes in the constraint graphs. As the maximal cliques algorithm runs in exponential time of the number of nodes, the inclusion of the filter decreases the chances that this part of the algorithm runs for more than one hour.

Remark 5.4. *The Iter as described in Section 5.1.2 can be incorporated in the recognition algorithm. This might lead to fewer and smaller SMSSPs that are found, but maybe also to SMSSPs that give better results in the exploitation. Furthermore it excludes negative processing times and might decrease the chance that the algorithm runs for over an hour.*

Furthermore, some additional information about the merged graph in which the maximal cliques are searched for was available for all instances. The merged graph of the instances on which the algorithm terminated within an hour consisted of not more than around 200.000 edges for all completed instances. For two cases that did not terminate within the time frame of an hour, this value was exceeded easily: the merged graph contained around 2.25 million edges in one case and around 9.49 million edges in the other case. It is likely that this amount of edges is (part of) the reason that the algorithm was not brought to an end within an hour. However, for the third case, the merged graph was in terms of number of nodes and number of edges smaller than some cases for which the algorithm ceased within an hour. From this, one may conclude that not only the size, but also the structure of the graph influences the amount of time that is necessary to find all maximal cliques.

Remark 5.5. *Not only size, but also graph structure in uences the running time of the FC part of the recognition algorithm.*

To complete the discussion of the results on the recognition algorithm, it can be said that the algorithm is efficient for the parts that in theory have a worst-case complexity that is polynomial in terms of the number of constraints and variables. For the part where the worst-case complexity is exponential, indeed it can happen that the running time becomes very large, although incorporating the filter as described in Section 5.1.2 might decrease the chance that this happens. As the results show, the implementation of the algorithm mainly consumes time for reading the model and for constructing the constraint graphs from the model. It is likely that this part of the algorithm takes less time when it is implemented in a MIP solver. The first reason is that a MIP solver loops through all the constraints already to find different kinds of possible cuts. Then, the additional check for a scheduling constraint could probably be incorporated efficiently in that part of MIP solving software. Secondly, this expected speed-up could be explained by the fact that these solvers in general are implemented in more efficient programming languages than Python, such as C or C++.

Remark 5.6. *The running time of the recognition algorithm potentially decreases even further when it is incorporated in MIP solver software.*

5.2 Exploitation of single machine scheduling subproblems

In this section the implementation details, results and its interpretation are given for the cut generation and exploitation of SMSSPs. This includes results on the amount of generated cuts and on solving various models with and without additional cuts.

5.2.1 Method

After the recognition algorithm has identified SMSSPs in various MIP models, the next step is to generate cutting planes and add them as constraints to the models. In this study, it was chosen to try to generate the release date and precedence cuts for the 9 MIP instances in which SMSSPs were found. These cutting planes are generated before a model is solved and are added to the model as constraints, hence creating a new model.

After cutting plane generation, resulting in models with and without additional constraints, the different models are tested to see whether the updated models are solved faster. Therefore, besides the original model, a copy with all generated cutting planes and, if applicable, copies with only one of the different generated sets of cuts were tested. The models are solved twice using Gurobi 9.1.1 [2] in Spyder 4.2.5 [25] on all six cores of an Intel Core i7 CPU running at 2.60 GHz with 16 GB RAM. First, the models are solved with the Gurobi functionalities *cuts*, *heuristics* and *presolve* turned off and with a time limit of one hour. Secondly, the models are solved with all Gurobi functionalities on default setting, but again with a time limit of one hour.

5.2.2 Results

Table 8: Number of original constraints in the model and generated cutting planes per type per MIP instance in which at least one SMSSP was found.

Instance	Original	Dominance	Release date	Precedence
ic97_potential	1046	0	88	0
neos-3046615-murg	498	9	256	0
neos-3656078-kumeu	17656	0	0	0
neos-3754224-navua	232387	0	240	0
piperout-08	14589	0	4	32
rd-rplusc-21	125899	0	0	0
supportcase40	38192	0	0	0
traininstance2	15603	0	12058	5524
traininstance6	12309	0	719	110

The first result is on the cutting plane generation. In Table 8 the number of non-trivial cuts per type are given for all instances where a SMSSP was found. From this table, one can see that actually only for 6 of 9 instances non-trivial cuts were generated. The set of dominance cuts is the least common to contain any non-trivial cuts, as only for the *neos-3046615-murg* instance cuts of this type are generated. For three instances non-trivial precedence cuts were calculated, while to six instances release date cuts could be added.

Table 9: Results of solving original and modified models without using Gurobi functionalities *cuts*, *heuristics* and *presolve*. Numbers with a red background indicate a larger solving time or a looser bound than the result of the original model, a green background indicates a smaller solving time or a tighter bound. Numbers with no background indicate an equal result.

	Original			Modified		
Instance	UB	LB	Time	UB	LB	Time
ic97_potential	3942	3914	3600	3942	3918	3600
neos-3046615-murg	1600	993	3600	1600	1600	< 1
neos-3754224-navua	inf	55688	3600	inf	55688	3600
piperout-08	125055	125055	15	125055	125055	49
traininstance2	75430	0	3600	75400	0	3600
traininstance6	28290	11490	3600	28649	6251	3600

The next results are on solving the models. First, in Tables 9 and 10 the results with Gurobi functionalities *cuts*, *heuristics* and *presolve* switched off are shown. From Table 9, it can be seen that only the modified instance *neos-3046615-murg* and both the original and modified instance *piperout-08* were solved by Gurobi within an hour. When comparing the results between the original and the modified model, one can observe that for the cases *ic97_potential*, *neos-3046615-murg* and *traininstance2* the gap between the lower and upper

Table 10: Results of solving models with single additional set of cutting planes without using *cuts*, *heuristics* and *presolve*. Numbers with a red background indicate a larger solving time or a looser bound than the result of the original model as given in Table 9, a green background indicates a smaller solving time or a tighter bound. Numbers with no background indicate an equal result.

	Dominance cuts			Release date cuts			Precedence cuts		
Instance	UB	LB	Time	UB	LB	Time	UB	LB	Time
neos-3046615-murg	1602	962	3600	1600	1600	< 1	N/A	N/A	N/A
piperout-08	N/A	N/A	N/A	125055	125055	26	125055	125055	56
traininstance2	N/A	N/A	N/A	74760	0	3600	74720	0	3600
traininstance6	N/A	N/A	N/A	28290	5040	3600	28290	7710	3600

bound is smaller for the modified instance, where for *neos-3046615-murg* it also holds true that the solving time is reduced. For the instances *neos-3754224-navua* and *piperout-08* the gaps are the same, although for the latter instance the solving time was larger for the modified model. For *traininstance6*, the gap between lower and upper bound is larger for the modified model.

From Table 10 it can be seen that models with additional release date cutting planes have the best results for all the instances compared to other single additional constraint set models. For the cases *neos-3046615-murg*, *traininstance2* and *traininstance6*, this follows from the smaller gap between lower and upper bound, while for the models *neos-3046615-murg* and *piperout-08* the solving time is smaller for the model with additional release date cuts.

Table 11: Results of solving original and modified model with default settings. Numbers with a red background indicate a larger solving time than the result of the original model, a green background indicates a smaller solving time. Numbers with no background indicate an equal result.

	Original			Modified		
Instance	UB	LB	Time	UB	LB	Time
ic97_potential	3942	3942	861	3942	3942	916
neos-3046615-murg	1600	1600	19	1600	1600	< 1
neos-3754224-navua	inf	56224	3600	inf	56224	3600
piperout-08	125055	125055	2	125055	125055	3
traininstance2	71820	71820	112	71820	71820	185
traininstance6	28290	28290	9	28290	28290	6

The following results are the results on solving models with Gurobi functionalities in standard options, but with a time limit of an hour. These results are given in Tables 11 and 12. In Table 11 it can be seen that Gurobi needs less time to find the optimal solution of the modified models *neos-3046615-murg* and *traininstance6*, while more time is needed to solve the models *ic97_potential*, *piperout-08* and *traininstance2*. For model *neos-3754224-navua*, the original and the modified model give the same results.

Table 12: Results of solving models with single additional set of cutting planes with default settings. Numbers with a red background indicate a larger solving time than the result of the original model as given in Table 11, a green background indicates a smaller solving time. Numbers with no background indicate an equal result.

	Dominance cuts			Release date cuts			Precedence cuts		
Instance	UB	LB	Time	UB	LB	Time	UB	LB	Time
neos-3046615-murg	1600	1600	17	1600	1600	1	N/A	N/A	N/A
piperout-08	N/A	N/A	N/A	125055	125055	6	125055	125055	3
traininstance2	N/A	N/A	N/A	71820	71820	157	71820	71820	115
traininstance6	N/A	N/A	N/A	28290	28290	6	28290	28290	8

In Table 12, it is shown that Gurobi solves the model with additional release date cuts the fastest for the instances *neos-3046615-murg* and *traininstance6*, remarkably the two instances for which the modified models are solved faster than the original models. For the cases *piperout-08* and *traininstance2*, Gurobi needs the smallest amount of time for the model with additional precedence cuts.

5.2.3 Analysis

The first component of the results on the exploitation of SMSSPs to reflect on is the method. In this study, it was chosen to generate all scheduling cuts before solving the models. To see whether the additional cuts could make any difference, the models where solved with and without those additional constraints. As described in Section 1.2, generating all possible cuts and add those to models is not how modern MIP solvers use cuts to solve MIP models faster. Instead, in every node of the B&B-tree a solver might spend a limited amount of time looking for cutting planes and thus will only add a limited number of cutting planes to the MIP. By choosing which cutting plane to add to the MIP, MIP solvers try to cut off the optimal solution in the most effective way, without losing too much time in solving future LP relaxations. By adding all conceivable cutting planes, it is guaranteed that any positive effect of a cutting plane is achieved, but this effect might be reduced or even wiped out by losing time on solving unnecessary complex LP relaxations.

Remark 5.7. *The scheduling cuts are not tested in the way current MIP solvers use cutting planes. When results show that exploiting scheduling cuts is advantageous, it should be tested whether it is maybe even more bene cial to use the cuts in the manner current MIP solvers would do.*

Per Table 9, in the case where Gurobi does not use the functionalities *cuts*, *heuristics* and *presolve*, the additional constraints show that there is potential for solving the models *ic97_potential*, *neos-3046615-murg* and *traininstance2* faster through the use of these cutting planes. However, as the functionalities

are activated again, it turns out that the instances *ic97_potential* and *traininstance2* are actually solved slower, as can be seen in Table 11. However, for *traininstance-6*, the modified model is solved slower without functionalities compared to the original model without functionalities, but faster when the functionalities are turned on again. All in all, this means that for four of six models for which cutting planes were generated there might be the possibility to solve the models faster using the structure of SMSSPs.

Remark 5.8. *For four of the tested instances, there is potential to decrease solving time using scheduling cuts.*

To be able to say something about how this potential can be cashed, it is good to take a better look at the results in Tables 9-12. In the case that some Gurobi functionalities are turned off, the gain achieved by the modified models *ic97_potential* and *neos-3046615-murg* can (mainly) be explained by the additional release date cuts, while for *traininstance2*, both models with an individual additional set of constraints induce a slightly better upper bound after one hour than the model that is modified with both additional constraint sets. For *piperout-08* and *traininstance6* the loss seems to be caused by both additional sets of release date cuts and precedence cuts.

In the case that Gurobi functionalities are turned on, the gain achieved by the modified models *neos-3046615-murg* and *traininstance6* seem to be primarily caused by the additional release date cuts, while it looks like that the added dominance and precedence cuts have a smaller effect on solving time. On the other hand, the delay in solving time for the modified models *ic97_potential*, *piperout-08* and *traininstance2* appear to be caused (mostly) by the release date cuts, while the extra precedence cuts give the impression of being less influential on slowing down the process of solving the MIP. Combining these findings with the results of the case that functionalities are turned off, it appears that the release date cuts are the most influential as well in speeding up as in slowing down the process of solving the MIP.

Remark 5.9. *The release date cuts seem to have to most e ect on the solving time, in positive sense as well as in negative sense.*

To hopefully be able to see how the cuts influence the process of solving the various MIP models, some additional experiments were performed. For all models, four experiments were executed: solve the model with a time limit of only one minute and solve only the root node of the branch-and-bound tree. Both these experiments are run with and without the Gurobi functionalities *cuts, heuristics* and *presolve*. The results of these experiments are not discussed in full detail here, but can be found in Appendix B.

From the tables in Appendix B it follows that for the case when the Gurobi functionalities are turned off, the improvement for *ic97_potential* and *neos-3046615-murg* is gained by having to explore fewer branch-and-bound nodes to get to the optimal solution. For *neos-3046615-murg* the root node quality has also improved as the lower bound is higher. For *traininstance2* a difference

in solution status can not be observed when only solving the root node or for one minute, so it is most likely that the improved upper bound after one hour is due to the fact that fewer branch-and-bound nodes have to be explored. For the instance *piperout-08*, the addition of release date cuts decreases the number of branch-and-bound nodes that need to be explored. However, it also induces more simplex iterations per branch-and-bound node and it requires more time per simplex iteration and as a result the time required to solve the model increases. The addition of precedence cuts to this model decreases the amount of simplex iterations per branch-and-bound node. On the other hand this addition increases the amount of nodes that need to be explored, causing the model to need more time to find the optimal solution. For *traininstance6* the solution status is not different when only solving the root node, but adding cuts does imply a larger upper bound after one minute of solving. This seems to be caused by needing more time to execute one simplex iteration, caused by the larger number of constraints. As the number of simplex iteration per branch-and-bound node hardly changes, this directly results into being able to explore fewer nodes in the same amount of time, thus having a larger upper bound after one minute.

In the case that the Gurobi functionalities are turned on, the addition of release date cuts causes the branch-and-bound algorithm to explore fewer nodes in order to find the optimal solution for the models *neos-3046615-murg* and *traininstance6*, resulting in a lower solving time. Especially for the model *neos-3046615-murg*, this improvement can be seen in root node quality as well, although the root nodes are solved a bit slower. Also adding the dominance cuts and precedence cuts do not seem to make a big difference for these models. For *traininstance2*, the root node quality is improved by adding only release date cuts, but for the other types of cuts and for the other models, the root node quality has become worse or the root node is solved slower. The one minute runs show that the gap between lower and upper bound is improved for the model *ic97_potential* by adding release date cuts. This can be seen as a surprise, as the modified model had a larger overall solving time than the original model. For *piperout-08* and *traininstance2*, adding one or both constraint sets slows the solving down or induces a larger gap between lower and upper bound after one minute. Remarkably, adding both sets of constraints gives better results than adding single constraint set for the model *piperout-08*.

Remark 5.10. *Most improvements in solving time seem to be the result of having to explore fewer nodes in the B&B-tree, but in some cases also the root node quality is improved.*

So, in general it can be said that the addition of constraints makes it possible to solve MIP models faster in some cases. Of the tested cuts, the most promising type of cuts are the release date cuts. However, simply adding sets of constraints does not always yield an improvement. For example, for *traininstance2* a total of 17582 cutting planes is generated and added to the model, while the original model contains only 15603 constraints. Also the number of non-zero coefficients increases from 41531 to 64637. This makes the model more difficult to solve, so it is not surprising that the solving time for the modified

model is increased compared to the solving time of the original model. As it is likely that a significant part of the additional cutting planes does not positively influence the solving time of the model, it would be interesting to see how the solving time would be affected when cutting planes are added to the model in more clever ways.

One of those more clever ways is to use the separation problem. At some point in the B&B-tree, an optimal solution with a variable that takes a non-integral value should be cut off. This can be done by finding a cut that is known to be true for the integral solution, but not for the found optimal solution. Any cut that satisfies these conditions can be selected, but preferably a cutting plane that cuts off an as large as possible part of the feasible region is chosen to be added to the set of constraints. Of course, single machine scheduling cuts are not the only possible cuts to select in a separation problem, but they can be part of the set of possible cuts.

Furthermore, it is possible to find additional criteria that could tell something more about when to generate cuts and apply them to MIP models. For example, does the percentage of constraints or variables of the MIP that are part of the SMSSPs indicate whether exploiting the SMSSPs is profitable? Another option is to check the amounts of arcs in the constraint graphs. The instance in which the most progress in terms of solving time was made in models with additional constraints, *neos-3046615-murg*, only has arcs in the *Regular Digraph*, while for instances where no progress was made, often only arcs in the *Precedence Digraph* are present. Therefore, it seems reasonable to only generate and use cutting planes of single machine scheduling problems where at least some arcs are present in the *Regular Digraph* D_r.

Remark 5.11. *The cutting planes could be added to the MIP in di erent ways than the used method, for example by solving the separation problem. Also the criterion that the SMSSP should contain non-precedence relations might be of use. Whether it is possible to base a measure on percentages of constraints or variables, requires further research.*

6 Discussion

To be able to draw some conclusions from this research, it is good to take one more look at the research question:

Research Question. *Can the presence of single machine scheduling subproblems in mixed integer programs be exploited to reduce the solving time of mixed integer programs?*

The brief and somewhat oversimplified answer to that question is yes. Using the developed recognition algorithm, it was clear that exploiting the SMSSP in the MIP instance *neos-3046615-murg* reduced the solving time. Also for the model *traininstance6* the model with additional cutting planes the solving time was decreased. Moreover, some other MIP instances showed the potential to profit from exploiting the SMSSP. However, for many of the other tested instances, the addition of the recognition algorithm to solving algorithms would increase the solving time as it only costs time to conclude that no SMSSP is present.

Naturally, during the course of the process of model selection, algorithm design, cut selection, implementation, tests and experiments, many choices and (partial) results are open for discussion or raise new questions. The most relevant aspects are addressed below.

The first facet of this discussion is the algorithm as presented in Section 3.3. As described in Section 5.1.1, some parts of the algorithm were modified before it was tested. Most significantly, an auxiliary graph to check (complex) relations between binary variables of complementary scheduling constraints was introduced. However, it turned out that no pair of binary variables was related through more than a single constraint, making the introduction of the auxiliary graph redundant.

Another important variation to the algorithm is to check whether the non-binary variables are completion or starting time variables. By checking which of those two types of variables is present instead of simply assuming one of the two, the processing times are likely to take larger values in case the assumption is wrong. This would then imply that the later generated cutting planes are stronger and therefore could lead to faster solving of MIPs that contain SMSSPs. Hence, it is beneficial to add this check to implementations of the algorithm.

Recommendation 1. *The inclusion of the auxiliary graph in the recognition algorithm to detect complex binary relations can be left out. The additional check for completion or starting time variables seems useful and it is recommended to incorporate this in the algorithm.*

Furthermore, in Section 5.1.2 a filter was introduced as it is very likely that small SMSSPs and SMSSPs with non-positive processing times will not generate cutting planes that cut off large parts of the solution space. A possible implementation is to not allow arcs that imply a processing time of 0 in the constraint graphs and to modify the subalgorithm that finds all maximal cliques. During the experiments the applied filter removed SMSSPs that contained only

two jobs. However, one may wonder what the minimal size is of a SMSSP that will generate useful cutting planes that help reducing the solving time of MIPs. In the tested library, the MIP instance *neos-3046615-murg* contains a SMSSP of size 16 for which the additional cuts made a big difference, while the instance *traininstance2* also contains SMSSPs of size 16 but the set of additional cutting planes slows the branch-and-cut procedure down. As the library does not contain a lot of MIPs with SMSSPs, it is difficult to give a reasoning that results in a minimal size of a SMSSP. As it is likely that SMSSPs consisting of three jobs are still too small to have a great impact on the solving time, the idea of exploiting only SMSSPs of a minimal size is useful.

Recommendation 2. *It is suggested to only include arcs that imply a processing time larger than 0 in the constraint graphs. Furthermore, a minimal number of jobs for SMSSPs is likely a useful addition too, but it requires further research to decide what this minimal number should be.*

A possible extension of the filter is to remove SMSSPs in which only precedence relations exist between the jobs in the subproblem. It seems reasonable that this kind of SMSSPs does not generate cutting planes that help solving MIPs a lot faster, as these subproblems do not contain constraints with binary variables with large coefficients, but only with integer or continuous variables. The results also show that models with SMSSPs with precedence relations only are not solved faster and this result justifies the elimination of this type of subproblems. This can be implemented in the algorithm by checking graph sizes after constructing the constraint graphs. Then, the algorithm can be terminated when there are no edges in the *Regular Digraph* D_r.

One way wonder as well whether useful cutting planes are generated from SMSSPs that largely have precedence relations between its jobs. In our experiments, it is in most cases not known which SMSSPs induce useful cutting planes and which properties those SMSSPs have, due to the design of the experiment. Therefore it can not be said what should be the minimal percentage of non-precedence relations in a SMSSP.

Recommendation 3. *SMSSPs with only precedence relations should be computed by the recognition algorithm. This is possibly also true for SMSSPs with large percentages of precedence relations, but it requires further research to determine a maximal percentage of precedence relations.*

The part of the recognition algorithm where it is most likely to lose more than one hour, is the part where are all maximal cliques are found. As an alternative for this part, the maximal cliques algorithm can be replaced by heuristics that run in polynomial time, but only find the largest maximal clique. This alternative could work well; it is likely that the most useful cutting planes are generated from the largest SMSSP as then the largest number of variables is involved in the cuts. However, it could be that combining cutting planes generated from different SMSSPs earns even better results, but it is unknown whether this yields a similar effect in terms of solving time as only exploiting cutting planes generated from the largest SMSSP.

Recommendation 4. *It requires further research to find out whether replacing the all maximal cliques algorithm by a largest maximal clique heuristic achieves a similar effect.*

A final remark about the implementation of the algorithm can be made about the programming language in which the algorithm is executed. For the type of task performed in the first part of the algorithm, programming languages like C or C++ are typically more efficient. By implementing the algorithm in such a programming language could decrease the running time of the algorithm.

Recommendation 5. *It is recommended to implement the algorithm in a programming language like C or C++ and perform the same tests to see whether this reduces the running time of the recognition algorithm.*

After running the algorithm, sets of cuts are selected to help solving MIPs faster. The selection criteria that an inequality should contain at least one variable that is present in the MIP and should not introduce new variables, are inescapable. However, the third criterion that limits the amount of cutting planes per type, might be dropped. Of course, it requires further research to see whether the cutting plane generation does not take too much time, as in that case the MIP solving takes more time instead of less time. Some examples of sets of valid inequalities that grow exponentially in the number of jobs in a scheduling problem can be found in the paper by Nemhauser and Savelsbergh [14].

Besides dropping selection criteria, there is another way to generate more different types of cutting planes. By reversing the time horizon of the scheduling problem, due date cuts can be derived from release date cuts. Therefore, the latest due date of any job may be seen as the starting point of the time horizon, while earlier due dates can be viewed as release dates. Of course, it requires finite due dates to be able to generate non-trivial cutting planes.

Recommendation 6. *It is recommended to seek for due date cuts, possibly derived from release date cuts, and sets of cuts that grow exponentially in the number of jobs in a scheduling problem. Additionally, it is suggested to perform experiments to see whether these cuts can be generated in little time and whether the cuts can help reducing solving time.*

Finally, the results on solving (modified) MIP models show that an improvement in solving time by adding single machine scheduling cutting planes is possible, but not in many cases. As only a few MIP instances in the MIPLIB benchmark set contained a SMSSP that uses natural date variables, it makes sense to use larger selections of MIP instances, such as the MIPLIB collection set. This then helps to generate more significant results.

Recommendation 7. *For future research, it is worth considering to perform tests on larger sets of MIP instances.*

One of the problems with the approach used in this thesis, is that the additional constraints can also cause an increased solving time. Although further

tests with the above suggested algorithm modifications and improved filters could lead to fewer models for which the solving time is incremented or to a smaller increase of the solving time, there is possibly also room for improvement in the way in which the generated cutting planes are exploited. Instead of simply adding all generated cutting planes to the model, the generation of scheduling cuts can be integrated in the separation problems that are solved during the process of branch-and-cut.

Recommendation 8. *It is recommended to perform experiments where the scheduling cutting planes are included in the separation problems to see whether solving time of MIP instances can be decreased even more.*

References

[1] A. H. Land and A. G. Doig, "An Automatic Method of Solving Discrete Programming Problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, Jul. 1960, ISSN: 00129682. DOI: 10.2307/1910129.

[2] Gurobi Optimization LLC, *Gurobi Optimizer Reference Manual*, 2021. [Online]. Available: https://www.gurobi.com.

[3] IBM Corporation, *IBM ILOG CPLEX Optimization Studio CPLEX User's Manual*, 2017.

[4] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a Large-Scale Traveling-Salesman Problem," *Operations Research*, vol. 2, no. 4, pp. 393–410, Nov. 1954, ISSN: 0096-3984. DOI: 10.1287/OPRE.2.4.393. [Online]. Available: https://pubsonline.informs.org/doi/abs/10.1287/opre.2.4.393.

[5] W. Smith, "Various optimizers for single-stage production," *Naval Research Logistics Quarterly*, vol. 3, no. 1-2, pp. 59–66, 1956.

[6] E. Balas, "On the facial structure of scheduling polyhedra," *Mathematical Programming*, no. 24, 1985. DOI: 10.1007/BFb0121051.

[7] M. Queyranne, "Structure of a simple scheduling polyhedron," *Mathematical Programming*, vol. 58, no. 1-3, Jan. 1993, ISSN: 0025-5610. DOI: 10.1007/BF01581271.

[8] M. Queyranne and Y. Wang, "Single-Machine Scheduling Polyhedra with Precedence Constraints," *Mathematics of Operations Research*, vol. 16, no. 1, Feb. 1991, ISSN: 0364-765X. DOI: 10.1287/moor.16.1.1.

[9] J. P. Sousa and L. A. Wolsey, "A time indexed formulation of non-preemptive single machine scheduling problems," *Mathematical Programming*, vol. 54, no. 1-3, Feb. 1992, ISSN: 0025-5610. DOI: 10.1007/BF01586059.

[10] J. van den Akker, C. van Hoesel, and M. Savelsbergh, "A polyhedral approach to single-machine scheduling problems," *Mathematical Programming*, vol. 85, no. 3, Aug. 1999, ISSN: 0025-5610. DOI: 10.1007/s10107990047a.

[11] K. Šorić, "A cutting plane algorithm for a single machine scheduling problem," *European Journal of Operational Research*, vol. 127, no. 2, Dec. 2000, ISSN: 03772217. DOI: 10.1016/S0377-2217(99)00493-2.

[12] M. E. Dyer and L. A. Wolsey, "Formulating the single machine sequencing problem with release dates as a mixed integer program," *Discrete Applied Mathematics*, vol. 26, no. 2-3, pp. 255–270, Mar. 1990, ISSN: 0166-218X. DOI: 10.1016/0166-218X(90)90104-K.

[13] J. Blazewicz, M. Dror, and J. Weglarz, "Mathematical programming formulations for machine scheduling: A survey," *European Journal of Operational Research*, vol. 51, no. 3, Apr. 1991, ISSN: 03772217. DOI: 10.1016/0377-2217(91)90304-E.

[14] G. L. Nemhauser and M. W. P. Savelsbergh, "A Cutting Plane Algorithm for the Single Machine Scheduling Problem with Release Times," in *Combinatorial Optimization: New Frontiers in the Theory and Practice*, M. Akgül, H. Hamacher, and S. Tufecki, Eds., vol. 82, Springer, Berlin, Heidelberg, 1992, pp. 63–83. DOI: 10.1007/978-3-642-77489-8{_}4.

[15] F. A. Chudak and D. S. Hochbaum, "A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine," *Operations Research Letters*, vol. 25, no. 5, Dec. 1999, ISSN: 01676377. DOI: 10.1016/S0167-6377(99)00056-5.

[16] J. B. Lasserre and M. Queyranne, "Generic Scheduling Polyhedra and a New Mixed-Integer Formulation for Single-Machine Scheduling," in *Proceedings of the 2nd IPCO (Integer Programming and Combinatorial Optimization) conference*, E. Balas, G. Cornuéjois, and R. Kannan, Eds., Carnegie Mellon University, May 1992, pp. 136–149. [Online]. Available: http://hdl.handle.net/2078.1/173068.

[17] S. Dauzere-Peres, "An efficient formulation for minimizing the number of late jobs in single-machine scheduling," in *1997 IEEE 6th International Conference on Emerging Technologies and Factory Automation Proceedings, EFTA '97*, IEEE, 1997, pp. 442–445, ISBN: 0-7803-4192-9. DOI: 10.1109/ETFA.1997.616311.

[18] M. Sevaux and S. Dauzère-Pérès, "Genetic algorithms to minimize the weighted number of late jobs on a single machine," *European Journal of Operational Research*, vol. 151, no. 2, Dec. 2003, ISSN: 03772217. DOI: 10.1016/S0377-2217(02)00827-5.

[19] K. Khowala, A. Keha, and J. Fowler, "A comparison of different formulations for the non-preemptive single machine total weighted tardiness scheduling problem," in *The Second Multidisciplinary International Conference on Scheduling: Theory & Application (MISTA)*, 2005.

[20] 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, Feb. 2009, ISSN: 0360-8352. DOI: 10.1016/J.CIE.2008.06.008.

[21] K.-C. Ying, C.-Y. Cheng, S.-W. Lin, and C.-Y. Hung, "Comparative Analysis of Mixed Integer Programming Formulations for Single-Machine and Parallel-Machine Scheduling Problems," *IEEE Access*, vol. 7, pp. 152 998–153 011, 2019. DOI: 10.1109/ACCESS.2019.2947685.

[22] S. Warshall, "A Theorem on Boolean Matrices," *Journal of the ACM*, vol. 9, no. 1, Jan. 1962, ISSN: 0004-5411. DOI: 10.1145/321105.321107.

[23] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments," *Theoretical Computer Science*, vol. 363, no. 1, pp. 28–42, Oct. 2006, ISSN: 0304-3975. DOI: 10.1016/J.TCS.2006.06.015.

[24] L. Wolsey, "Formulating single machine scheduling problems with precedence constraints," in *Economic Decision Making: Games, Econometrics and Optimisation*, J. Gabszewicz, J.-F. Richard, and L. Wolsey, Eds., Amsterdam, 1990, pp. 473–484.

[25] P. Raybaut, "Spyder-documentation," *Available online at: pythonhosted.org*, 2009.

[26] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009, ISBN: 1441412697.

[27] A. Gleixner, G. Hendel, G. Gamrath, T. Achterberg, M. Bastubbe, T. Berthold, P. Christophel, K. Jarck, T. Koch, J. Linderoth, M. Lübbecke, H. D. Mittelmann, D. Ozyurt, T. K. Ralphs, D. Salvagnin, and Y. Shinano, "MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library," *Mathematical Programming Computation*, vol. 13, no. 3, Sep. 2021, ISSN: 1867-2949. DOI: 10.1007/s12532-020-00194-3.

[28] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference (sciPy 2008)*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, Aug. 2008, pp. 11–15.

[29] R. Bellman, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, no. 1, Apr. 1958, ISSN: 0033-569X. DOI: 10.1090/qam/102435.

A Results of the test of recognition algorithm

Table 13: Detailed results of the test of the recognition algorithm.

	Time						Original		Filtered	
Instance	Total	CG	TC	CB	FC	CM	Cliques	max	Cliques	max
30n20b8	< 1	< 1	< 1	< 1	< 1	< 1	60	2	0	-
50v-10	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
academictimetablesmall	3	3	< 1	< 1	< 1	< 1	0	-	0	-
app1-2	7	7	< 1	< 1	< 1	< 1	13300	2	0	-
assign1-5-8	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
atlanta-ip	3	3	< 1	< 1	< 1	< 1	0	-	0	-
b1c1s1	< 1	< 1	< 1	< 1	< 1	< 1	16	2	0	-
bab2	4	4	< 1	< 1	< 1	< 1	0	-	0	-
bab6	5	5	< 1	< 1	< 1	< 1	0	-	0	-
beasleyC3	< 1	< 1	< 1	< 1	< 1	< 1	160	2	0	-
binkar10_1	< 1	< 1	< 1	< 1	< 1	< 1	180	2	0	-
blp-ar98	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
blp-ic98	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
bnatt400	1	1	< 1	< 1	< 1	< 1	0	-	0	-
bnatt500	1	1	< 1	< 1	< 1	< 1	0	-	0	-
bppc4-08	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
brazil3	2	2	< 1	< 1	< 1	< 1	0	-	0	-
buildingenergy	35	33	1	< 1	1	1	59964	3	0	-
cbs-cta	1	1	< 1	< 1	< 1	< 1	0	-	0	-
chromaticindex1024-7	8	8	< 1	< 1	< 1	< 1	0	-	0	-
chromaticindex512-7	4	4	< 1	< 1	< 1	< 1	0	-	0	-
cmflsp50-24-8-8	1	1	< 1	< 1	< 1	< 1	0	-	0	-

CMS750_4	> 3600	2	9	< 1	-	-	-	-	-	-
co-100	2	2	< 1	< 1	< 1	< 1	0	-	0	-
cod105	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
comp07-2idx	3	3	< 1	< 1	< 1	< 1	0	-	0	-
comp21-2idx	2	2	< 1	< 1	< 1	< 1	0	-	0	-
cost266-UUE	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
cryptanalysiskb128n5obj14	12	12	< 1	< 1	< 1	< 1	0	-	0	-
cryptanalysiskb128n5obj16	12	12	< 1	< 1	< 1	< 1	0	-	0	-
csched007	< 1	< 1	< 1	< 1	< 1	< 1	1	3	0	-
csched008	< 1	< 1	< 1	< 1	< 1	< 1	51	11	0	-
cvs16r128-89	1	1	< 1	< 1	< 1	< 1	0	-	0	-
dano3_3	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
dano3_5	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
decomp2	1	1	< 1	< 1	< 1	< 1	0	-	0	-
drayage-100-23	1	1	< 1	< 1	< 1	< 1	0	-	0	-
drayage-25-23	1	1	< 1	< 1	< 1	< 1	0	-	0	-
dws008-01	1	1	< 1	< 1	< 1	< 1	28	2	0	-
eil33-2	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
eilA101-2	1	1	< 1	< 1	< 1	< 1	0	-	0	-
enlight_hard	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
ex10	9	9	< 1	< 1	< 1	< 1	0	-	0	-
ex9	5	5	< 1	< 1	< 1	< 1	0	-	0	-
exp-1-500-5-5	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
fast0507	1	1	< 1	< 1	< 1	< 1	0	-	0	-
fastxgemm-n2r6s0t2	1	1	< 1	< 1	< 1	< 1	1224	2	0	-
fhnw-binpack4-4	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
fhnw-binpack4-48	1	1	< 1	< 1	< 1	< 1	0	-	0	-
fiball	1	1	< 1	< 1	< 1	< 1	43	6	0	-

gen-ip002	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
gen-ip054	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
germanrr	1	1	< 1	< 1	< 1	< 1	0	-	0	-
gfd-schedulen180f7d50m30k18	> 3600	56	1	< 1	-	-	-	-	-	-
glass-sc	1	1	< 1	< 1	< 1	< 1	0	-	0	-
glass4	< 1	< 1	< 1	< 1	< 1	< 1	9	2	0	-
gmu-35-40	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
gmu-35-50	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
graph20-20-1rand	1	1	< 1	< 1	< 1	< 1	0	-	0	-
graphdraw-domain	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
h80x6320d	1	1	< 1	< 1	< 1	< 1	0	-	0	-
highschool1-aigio	15	15	< 1	< 1	< 1	< 1	0	-	0	-
hypothyroid-k1	1	1	< 1	< 1	< 1	< 1	0	-	0	-
ic97_potential	< 1	< 1	< 1	< 1	< 1	< 1	251	6	3	6
icir97_tension	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
irish-electricity	13	13	< 1	< 1	< 1	< 1	5184	2	0	-
irp	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
istanbul-no-cutoff	3	3	< 1	< 1	< 1	< 1	5	2	0	-
k1mushroom	3	3	< 1	< 1	< 1	< 1	0	-	0	-
lectsched-5-obj	5	5	< 1	< 1	< 1	< 1	29	4	0	-
leo1	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
leo2	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
lotsize	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
mad	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
map10	43	38	1	< 1	1	2	13449	40	0	-
map16715-04	43	39	1	< 1	1	2	13449	40	0	-
markshare2	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
markshare_4_0	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-

mas74	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
mas76	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
mc11	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
mcsched	< 1	< 1	< 1	< 1	< 1	< 1	2	2	0	-
mik-250-20-75-4	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
milo-v12-6-r2-40-1	1	1	< 1	< 1	< 1	< 1	2	2	0	-
momentum1	5	5	< 1	< 1	< 1	< 1	179	2	0	-
mushroom-best	1	1	< 1	< 1	< 1	< 1	113	2	0	-
mzzv11	1	1	< 1	< 1	< 1	< 1	0	-	0	-
mzzv42z	1	1	< 1	< 1	< 1	< 1	0	-	0	-
n2seq36q	1	1	< 1	< 1	< 1	< 1	0	-	0	-
n3div36	1	1	< 1	< 1	< 1	< 1	0	-	0	-
n5-3	< 1	< 1	< 1	< 1	< 1	< 1	144	2	0	-
n9-3	< 1	< 1	< 1	< 1	< 1	< 1	176	2	0	-
neos-1122047	> 3600	7	57	< 1	-	-	-	-	-	-
neos-1171448	2	2	< 1	< 1	< 1	< 1	0	-	0	-
neos-1171737	1	1	< 1	< 1	< 1	< 1	0	-	0	-
neos-1354092	1	1	< 1	< 1	< 1	< 1	0	-	0	-
neos-1445765	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
neos-1456979	1	1	< 1	< 1	< 1	< 1	0	-	0	-
neos-1582420	1	1	< 1	< 1	< 1	< 1	0	-	0	-
neos-2075418-temuka	46	46	< 1	< 1	< 1	< 1	0	-	0	-
neos-2657525-crna	< 1	< 1	< 1	< 1	< 1	< 1	57	3	0	-
neos-2746589-doon	4	4	< 1	< 1	< 1	< 1	0	-	0	-
neos-2978193-inde	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
neos-3004026-krka	2	2	< 1	< 1	< 1	< 1	0	-	0	-
neos-3024952-loue	1	< 1	< 1	< 1	< 1	< 1	3075	2	0	-
neos-3046615-murg	< 1	< 1	< 1	< 1	< 1	< 1	1	16	1	16

neos-3083819-nubu	1	1	< 1	< 1	< 1	< 1	4192	3	0	-
neos-3216931-puriri	1	1	< 1	< 1	< 1	< 1	0	-	0	-
neos-3381206-awhea	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
neos-3402294-bobin	67	67	< 1	< 1	< 1	< 1	0	-	0	-
neos-3402454-bohle	331	331	< 1	< 1	< 1	< 1	0	-	0	-
neos-3555904-turama	17	17	< 1	< 1	< 1	< 1	0	-	0	-
neos-3627168-kasai	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
neos-3656078-kumeu	2	2	< 1	< 1	< 1	< 1	330	4	7	4
neos-3754224-navua	29	27	1	< 1	< 1	< 1	16380	5	2355	5
neos-3754480-nidda	< 1	< 1	< 1	< 1	< 1	< 1	100	2	0	-
neos-3988577-wolgan	5	5	< 1	< 1	< 1	< 1	0	-	0	-
neos-4300652-rahue	9	9	< 1	< 1	< 1	< 1	3458	2	0	-
neos-4338804-snowy	< 1	< 1	< 1	< 1	< 1	< 1	21	2	0	-
neos-4387871-tavua	1	1	< 1	< 1	< 1	< 1	0	-	0	-
neos-4413714-turia	2	2	< 1	< 1	< 1	< 1	0	-	0	-
neos-4532248-waihi	20	20	< 1	< 1	< 1	< 1	0	-	0	-
neos-4647030-tutaki	4	4	< 1	< 1	< 1	< 1	0	-	0	-
neos-4722843-widden	13	13	< 1	< 1	< 1	< 1	2668	2	0	-
neos-4738912-atrato	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
neos-4763324-toguru	13	13	< 1	< 1	< 1	< 1	0	-	0	-
neos-4954672-berkel	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
neos-5049753-cuanza	39	39	< 1	< 1	< 1	< 1	0	-	0	-
neos-5052403-cygnet	8	8	< 1	< 1	< 1	< 1	0	-	0	-
neos-5075914-elvire	< 1	< 1	< 1	< 1	< 1	< 1	12	2	0	-
neos-5093327-huahum	8	7	< 1	< 1	< 1	< 1	19068	4	0	-
neos-5104907-jarama	59	58	1	< 1	1	< 1	26208	2	0	-
neos-5107597-kakapo	1	1	< 1	< 1	< 1	< 1	57	6	0	-
neos-5114902-kasavu	113	113	< 1	< 1	< 1	< 1	0	-	0	-

neos-5188808-nattai	4	4	< 1	< 1	< 1	< 1	4752	3	0	-
neos-5195221-niemur	5	5	< 1	< 1	< 1	< 1	0	-	0	-
neos-631710	20	20	< 1	< 1	< 1	< 1	0	-	0	-
neos-662469	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
neos-787933	1	1	< 1	< 1	< 1	< 1	0	-	0	-
neos-827175	2	2	< 1	< 1	< 1	< 1	84	2	0	-
neos-848589	2	2	< 1	< 1	< 1	< 1	0	-	0	-
neos-860300	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
neos-873061	11	11	< 1	< 1	< 1	< 1	801	2	0	-
neos-911970	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
neos-933966	2	2	< 1	< 1	< 1	< 1	0	-	0	-
neos-950242	4	4	< 1	< 1	< 1	< 1	0	-	0	-
neos-957323	1	1	< 1	< 1	< 1	< 1	0	-	0	-
neos-960392	1	1	< 1	< 1	< 1	< 1	0	-	0	-
neos17	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
neos5	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
neos8	6	6	< 1	< 1	< 1	< 1	0	-	0	-
neos859080	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
net12	2	2	< 1	< 1	< 1	< 1	0	-	0	-
netdiversion	15	15	< 1	< 1	< 1	< 1	0	-	0	-
nexp-150-20-8-5	1	1	< 1	< 1	< 1	< 1	0	-	0	-
ns1116954	16	16	< 1	< 1	< 1	< 1	0	-	0	-
ns1208400	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
ns1644855	7	6	< 1	< 1	< 1	< 1	10000	2	0	-
ns1760995	71	71	< 1	< 1	< 1	< 1	132	3	0	-
ns1830653	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
ns1952667	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
nu25-pr12	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-

nursesched-medium-hint03	2	2	< 1	< 1	< 1	< 1	0	-	0	-
nursesched-sprint02	1	1	< 1	< 1	< 1	< 1	0	-	0	-
nw04	1	1	< 1	< 1	< 1	< 1	0	-	0	-
opm2-z10-s4	18	18	< 1	< 1	< 1	< 1	0	-	0	-
p200x1188c	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
peg-solitaire-a3	1	1	< 1	< 1	< 1	< 1	0	-	0	-
pg	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
pg5_34	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
physicianshed3-3	31	31	< 1	< 1	< 1	< 1	1332	2	0	-
physicianshed6-2	20	20	< 1	< 1	< 1	< 1	116	2	0	-
piperout-08	2	2	< 1	< 1	< 1	< 1	108	3	68	3
pk1	< 1	< 1	< 1	< 1	< 1	< 1	30	2	0	-
proteindesign121hz512p9	1	1	< 1	< 1	< 1	< 1	0	-	0	-
proteindesign122trx11p8	1	1	< 1	< 1	< 1	< 1	0	-	0	-
qap10	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
radiationm18-12-05	6	5	< 1	< 1	< 1	< 1	3672	4	0	-
radiationm40-10-02	24	22	1	< 1	1	1	15600	4	0	-
rail01	6	6	< 1	< 1	< 1	< 1	0	-	0	-
rail02	13	13	< 1	< 1	< 1	< 1	0	-	0	-
rail507	1	1	< 1	< 1	< 1	< 1	0	-	0	-
ran14x18-disj-8	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
rd-rplusc-21	15	15	< 1	< 1	< 1	< 1	6	22	1	20
reblock115	1	1	< 1	< 1	< 1	< 1	0	-	0	-
rmatr100-p10	1	1	< 1	< 1	< 1	< 1	0	-	0	-
rmatr200-p5	5	5	< 1	< 1	< 1	< 1	0	-	0	-
rocI-4-11	1	1	< 1	< 1	< 1	< 1	4	11	0	-
rocII-5-11	3	3	< 1	< 1	< 1	< 1	6	11	0	-
rococoB10-011000	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-

rococoC11-011100	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
roi2alpha3n4	1	1	< 1	< 1	< 1	< 1	0	-	0	-
roi5alpha10n8	3	3	< 1	< 1	< 1	< 1	0	-	0	-
roll3000	< 1	< 1	< 1	< 1	< 1	< 1	1	2	0	-
s100	5	5	< 1	< 1	< 1	< 1	0	-	0	-
s250r10	4	4	< 1	< 1	< 1	< 1	0	-	0	-
satellites2-40	3	3	< 1	< 1	< 1	< 1	20	2	0	-
satellites2-60-fs	2	2	< 1	< 1	< 1	< 1	20	2	0	-
savsched1	37	37	< 1	< 1	< 1	< 1	2	2	0	-
sct2	< 1	< 1	< 1	< 1	< 1	< 1	68	2	0	-
seymour	1	1	< 1	< 1	< 1	< 1	0	-	0	-
seymour1	1	1	< 1	< 1	< 1	< 1	0	-	0	-
sing326	6	6	< 1	< 1	< 1	< 1	0	-	0	-
sing44	7	7	< 1	< 1	< 1	< 1	0	-	0	-
snp-02-004-104	17	15	< 1	< 1	< 1	< 1	21334	2	0	-
sorrell3	19	19	< 1	< 1	< 1	< 1	0	-	0	-
sp150x300d	< 1	< 1	< 1	< 1	< 1	< 1	84	2	0	-
sp97ar	1	1	< 1	< 1	< 1	< 1	0	-	0	-
sp98ar	1	1	< 1	< 1	< 1	< 1	0	-	0	-
splice1k1	2	2	< 1	< 1	< 1	< 1	0	-	0	-
square41	16	16	< 1	< 1	< 1	< 1	0	-	0	-
square47	29	29	< 1	< 1	< 1	< 1	0	-	0	-
supportcase10	19	19	< 1	< 1	< 1	< 1	0	-	0	-
supportcase12	25	22	1	< 1	1	1	45226	2	0	-
supportcase18	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
supportcase19	11	11	< 1	< 1	< 1	< 1	0	-	0	-
supportcase22	32	32	< 1	< 1	< 1	< 1	0	-	0	-
supportcase26	< 1	< 1	< 1	< 1	< 1	< 1	1	40	0	-

supportcase33	3	3	< 1	< 1	< 1	< 1	0	-	0	-
supportcase40	7	5	< 1	< 1	< 1	1	20020	6	3507	6
supportcase42	3	3	< 1	< 1	< 1	< 1	1	2	0	-
supportcase6	1	1	< 1	< 1	< 1	< 1	0	-	0	-
supportcase7	3	3	< 1	< 1	< 1	< 1	0	-	0	-
swath1	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
swath3	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
tbfp-network	1	1	< 1	< 1	< 1	< 1	0	-	0	-
thor50dday	7	7	< 1	< 1	< 1	< 1	0	-	0	-
timtab1	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
toll-like	1	1	< 1	< 1	< 1	< 1	0	-	0	-
tr12-30	< 1	< 1	< 1	< 1	< 1	< 1	12	2	0	-
traininstance2	2	2	< 1	< 1	< 1	< 1	2709	16	208	16
traininstance6	2	2	< 1	< 1	< 1	< 1	2034	11	33	11
trento1	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
triptim1	2	2	< 1	< 1	< 1	< 1	13	3	0	-
uccase12	16	15	< 1	< 1	< 1	< 1	4034	7	0	-
uccase9	7	6	< 1	< 1	< 1	< 1	2354	7	0	-
uct-subprob	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-
unitcal_7	6	6	< 1	< 1	< 1	< 1	7716	2	0	-
var-smallemery-m6j6	2	2	< 1	< 1	< 1	< 1	0	-	0	-
wachplan	< 1	< 1	< 1	< 1	< 1	< 1	0	-	0	-

B Results of additional experiments on exploitation

Table 14: Results for solving models for only one minute with the Gurobi functionalities *cuts*, *heuristics* and *presolve* off. A model with additional release date cuts is indicate by ^{y}, ^{x} denotes a model with additional precedence cuts and represents a model with additional dominance cuts.

Model	Time (s)	UB	LB	Gap	Number of expl. nodes	Number of simpl. it.	Simpl. it. per expl. node	Time per simpl. it. (μs)	Time per expl. node (ms)
ic97_potential	60	3950	3887	1.59%	621033	8605329	13.86	6.97	0.097
ic97_potential^{y}	60	3946	3889	1.44%	593090	8652954	14.59	6.93	0.101
neos-3046615-murg	60	1610	686	57.39%	3050805	6483341	2.13	9.25	0.020
neos-3046615-murg	60	1604	651	59.41%	2727952	5869295	2.15	10.22	0.022
neos-3046615-murg^{y}	1.3	1600	1600	0%	6910	82898	12.00	15.68	0.188
neos-3046615-murg ^{y}	0.94	1600	1600	0%	5773	61429	10.64	15.30	0.163
neos-3754224-navua	60	-	-	-	0	86168	-	696	-
neos-3754224-navua^{y}	60	-	-	-	0	130399	-	460	-
piperout-08	6.05	125055	125055	0%	2983	52035	17.44	116	2.028
piperout-08^{y}	18.75	125055	125055	0%	1963	59865	30.50	313	9.552
piperout-08^{x}	32.39	125055	125055	0%	8630	145652	16.88	222	3.753
piperout-08^{yx}	32.76	125055	125055	0%	6007	112943	18.80	290	5.454
traininstance2	60	-	0	-	13574	1110042	81.78	54.05	4.420
traininstance2^{y}	60	-	0	-	8727	258112	29.58	232	6.875
traininstance2^{x}	60	-	0	-	16086	471519	29.31	127	3.730
traininstance2^{yx}	60	-	0	-	4572	267634	58.54	224	13.123
traininstance6	60	29530	0	100%	16038	371143	23.14	162	3.741
traininstance6^{y}	60	31150	0	100%	14384	351641	24.45	171	4.171
traininstance6^{x}	60	29610	0	100%	17252	361441	20.95	166	3.478
traininstance6^{yx}	60	31830	0	100%	15055	297520	19.76	202	3.985

Table 15: Results for solving only the root node of the B&B-tree of the models with the Gurobi functionalities *cuts*, *heuristics* and *presolve* on. A model with additional release date cuts is indicate by ^{y}, ^{x} denotes a model with additional precedence cuts and represents a model with additional dominance cuts.

Model	Time (s)	UB	LB	Gap	Number of simplex iterations	Time per simplex iteration (ms)
ic97_potential	0.08	-	3868	-	1828	0.044
ic97_potential^{y}	0.08	-	3868	-	1564	0.051
neos-3046615-murg	0,02	-	192	-	309	0.065
neos-3046615-murg	0.02	-	192	-	353	0.057
neos-3046615-murg^{y}	0.04	-	1558	-	201	0.199
neos-3046615-murg ^{y}	0.05	-	1556	-	206	0.243
neos-3754224-navua	516.88	-	55687	-	231364	2.234
neos-3754224-navua^{y}	546.64	-	55688	-	231913	2.357
piperout-08	2.15	-	124955	-	10828	0.199
piperout-08^{y}	1.40	-	116353	-	8105	0.173
piperout-08^{x}	1.61	-	124955	-	8751	0.184
piperout-08^{yx}	1.50	-	124955	-	8982	0.167
traininstance2	0.52	-	0	-	3144	0.165
traininstance2^{y}	1.64	-	0	-	5836	0.281
traininstance2^{x}	0.65	-	0	-	3209	0.203
traininstance2^{yx}	0.82	-	0	-	3104	0.264
traininstance6	0.99	-	0	-	3303	0.300
traininstance6^{y}	0.70	-	0	-	3651	0.192
traininstance6^{x}	0.87	-	0	-	3915	0.222
traininstance6^{yx}	0.61	-	0	-	2986	0.204

Table 16: Results for solving models for only one minute with the Gurobi functionalities *cuts*, *heuristics* and *presolve* on. A model with additional release date cuts is indicate by ^{y}, ^{x} denotes a model with additional precedence cuts and represents a model with additional dominance cuts.

Model	Time (s)	UB	LB	Gap	Number of expl. nodes	Number of simpl. it.	Simpl. it. per expl. node	Time per simpl. it. (μs)	Time per expl. node (ms)
ic97_potential	60	3942	3931	0.29%	36458	2078728	57.02	28.86	1.65
ic97_potential^{y}	60	3942	3933	0.24%	36445	1802473	49.46	33.29	1.65
neos-3046615-murg	4.15	1600	1600	0%	15538	229428	14.77	18.09	0.27
neos-3046615-murg	3.26	1600	1600	0%	14112	171358	12.14	19.02	0.23
neos-3046615-murg^{y}	0.49	1600	1600	0%	1	550	550	890.91	490
neos-3046615-murg ^{y}	0.50	1600	1600	0%	1	614	614	814.33	500
neos-3754224-navua	60	-	-3722241826	-	0	0	-	-	-
neos-3754224-navua^{y}	60	-	-3722241826	-	0	0	-	-	-
piperout-08	1.46	125055	125055	0%	1	6879	6879	212.24	1460
piperout-08^{y}	6.63	125055	125055	0%	1	15400	15400	430.52	6630
piperout-08^{x}	3.62	125055	125055	0%	1	11999	11999	301.69	3620
piperout-08^{yx}	3.24	125055	125055	0%	1	10969	10969	295.38	3240
traininstance2	60	71820	69793	2.82%	254971	1848117	7.25	32.47	0.24
traininstance2^{y}	60	71820	69425	3.33%	263029	1740374	6.62	34.48	0.23
traininstance2^{x}	60	71920	69764	3.00%	183799	1616017	8.79	37.13	0.33
traininstance2^{yx}	60	72200	68872	4.61%	166411	1387425	8.34	43.25	0.36
traininstance6	3.43	28290	28290	0%	15243	91759	6.02	37.38	0.23
traininstance6^{y}	2.90	28290	28290	0%	11054	67222	6.08	43.14	0.26
traininstance6^{x}	2.77	28290	28290	0%	15296	78082	5.10	35.48	0.18
traininstance6^{yx}	2.90	28290	28290	0%	12466	83699	6.71	34.65	0.23

Table 17: Results for solving only the root node of the B&B-tree of the models with the Gurobi functionalities *cuts*, *heuristics* and *presolve* on. A model with additional release date cuts is indicate by ^{y}, ^{x} denotes a model with additional precedence cuts and represents a model with additional dominance cuts.

Model	Time (s)	UB	LB	Gap	Number of simplex iterations	Time per simplex iteration (ms)
ic97_potential	0.74	4065	3912	3.77%	4725	0.16
ic97_potential^{y}	0.55	4078	3905	4.25%	3893	0.14
neos-3046615-murg	0.16	1639	1554	5.19%	441	0.36
neos-3046615-murg	0.19	1639	1559	4.88%	506	0.38
neos-3046615-murg^{y}	0.55	1600	1600	0%	550	1.00
neos-3046615-murg ^{y}	0.52	1600	1600	0%	614	0.85
neos-3754224-navua	349.34	-	56045	-	189917	1.84
neos-3754224-navua^{y}	361.21	-	56045	-	189917	1.90
piperout-08	1.46	125055	125055	0%	6879	0.21
piperout-08^{y}	6.53	125055	125055	0%	15400	0.42
piperout-08^{x}	3.52	125055	125055	0%	11999	0.29
piperout-08^{yx}	3.22	125055	125055	0%	10969	0.29
traininstance2	0.32	79590	1	100%	722	0.44
traininstance2^{y}	0.39	78510	2610	96.68%	939	0.42
traininstance2^{x}	0.58	80800	1	100%	2370	0.24
traininstance2^{yx}	0.51	86860	1	100%	1828	0.28
traininstance6	0.28	30030	1	100%	603	0.46
traininstance6^{y}	0.45	32860	1	100%	1096	0.41
traininstance6^{x}	0.29	36180	1860	94.86%	397	0.73
traininstance6^{yx}	0.30	32550	2100	93.55%	380	0.79