MASTER

Mixed integer programming and constraint programming for production planning and scheduling at AC

Stavropoulos, A.

Award date:
2012

Link to publication
Mixed Integer Programming and Constraint Programming for Production Planning and Scheduling at AC

Master Thesis
Version 1.2 (public)
October 8, 2012

Student:
A. Stavropoulos

Supervisors:
prof. dr. ir. W.P.M. Nuijten
ir. J.W.M. Welberg
Preface

This master thesis constitutes the completion of my Master studies on Embedded Systems at Eindhoven University of Technology (TU/e). Furthermore, it concludes my internship performed in AC\(^1\) as part of a larger collaboration with TU/e. I really appreciate many people that helped me during this project, and I would like to thank them.

Above all, I would like to thank my graduation supervisor Wim Nuijten, for offering me the chance to do my graduation project at AC. His constant guidance and support helped me to get through any difficulties of the project and eventually reach our goals. Furthermore, his advice and constructive feedback was very important for the progress of the project as well as my personal development. I would also like to thank my supervisor at AC, Jan-Willem Welberg, for all the information and valuable feedback that he provided me during this project. His patience and trust on the project was very essential. In addition, I would like to thank my predecessors Roel Coset and Joost van Twist, for the clear description of the work that they have done on this project before me. Using their work as a starting point proved to be very helpful. Also Joost’s experience and advice played an important role. Furthermore, I would like to thank everyone else at AC for welcoming me at the company and contributing or showing interest to my project.

Finally, I feel grateful to my family for their support and for giving me the opportunity to study at TU/e, and to my friends for all the help and great time during my studies.

Alexandros Stavropoulos
Eindhoven, October 2012

\(^{1}\)The name of the company is anonymized in this public version of the document, to protect confidential information.
Abstract

In this document, a solution is proposed for modeling and solving the planning and scheduling problem at AC using Mixed Integer Programming (MIP) and Constraint Programming (CP). A previously developed solution by [6] and further improved by [29], was already available in the form of an automated planning tool developed using IBM ILOG Plant PowerOps (PPO) [14]. Although this solution gave great results and achieved very promising cost reductions regarding the availability of products for sales, it was considered to not be usable in practice mainly because the results were not reproducible and the solution method had a high variability. This meant that running the solution method twice from the same start situation, would give two different results that in addition could have significantly different quality. To improve on that and come up with models that AC would consider to be usable in practice, we set out to develop well-performing deterministic models. This thesis reports on the success of this endeavor and presents two completely new deterministic models that were developed using the previous solution method as a starting point.

The two models are developed using different approaches. The first is based on Mixed Integer Programming and proves to be unable to handle the high complexity of the problem found in AC. The second model though, based on Constraint Programming, is capable to handle this complexity and performs well both on small test instances as on real instances from AC’s practice. This is achieved after enhancing the original model with several improvements and modifications. In addition, a post processing method comprising three consecutive steps is developed for the CP model, improving the obtained results even further.

Several tests are performed to investigate the behavior of the CP model. At first six validation cases are tested, identical to the cases presented in [29]. The results indicate that the current model gives solutions of higher quality. Unfortunately, besides this comparison, a direct comparison of the results obtained by the CP model and the previously developed model is not possible, as several constraints of AC have changed since the development of the model proposed in [29]. Besides the validation cases as proposed in [29], other tests include the use of various objective weights, different running times and alternative objectives, or the relaxation of particular constraints to test how these affect the model performance. A disadvantage of the current solution is that the model cannot handle efficiently cases were the objectives of non delivery cost and inventory deficit cost are of equal or similar importance. More specifically, it is shown that the model performs better when the weight of non delivery cost is much higher than the weight of inventory deficit cost. Another characteristic of the current model solutions is that the total changeover costs are quite high, despite the improvement achieved by the post processing method. This outcome confirms the result of [29], where changeovers were high, suggesting that a higher number of changeovers may result in a more efficient use of the resources in the production facility of AC. Finally, the model results are compared to a manual solution to investigate whether the model gives solutions that can be used in practice by AC. The comparison indicates that the model outperforms the manual schedule, as the total costs are decreased by approximately 38%.
# Contents

1 Introduction  
1.1 Constraint Programming ............................................. 2  
1.2 Linear Programming and Mixed Integer Programming ............. 3  
1.3 Mixed Integer Programming and Constraint Programming Integration .... 5  
1.4 Alternative Approaches ............................................. 6

2 Problem Description  
2.1 The Planning and Scheduling Problem ............................... 9  
2.2 Changeovers ............................................................ 9  
2.3 Technical Constraints ................................................ 9  
2.4 Problem Input ......................................................... 10  
2.5 Problem Definition ................................................... 12

3 Mixed Integer Programming Model  
3.1 Input ................................................................. 13  
3.2 Output ............................................................... 14  
3.3 Constraints .......................................................... 15  
3.4 Objective Function .................................................... 21  
3.5 Alternative Modeling of Constraints .................................. 23  
3.6 Model Evaluation ...................................................... 25  
3.6.1 1 Product, 1 Production Line .................................... 26  
3.6.2 2 Products, 1 Production Line .................................... 27  
3.6.3 2 Products, 3 Production Lines .................................... 28  
3.6.4 April, May, June 2011 Data Instance .............................. 28

4 Constraint Programming Model  
4.1 Input ................................................................. 31  
4.2 Output ............................................................... 31  
4.3 Constraints .......................................................... 32  
4.4 Objective Function .................................................... 35  
4.5 Alternative Modeling and Improvement Modifications ............. 36  
4.5.1 Model decomposition .............................................. 37  
4.5.2 Search phases ...................................................... 39  
4.5.3 Maximum inventory level ......................................... 41  
4.5.4 Beginning of the changeover task intervals ..................... 42  
4.5.5 Initial changeovers ................................................. 44  
4.5.6 Search space reduction .......................................... 44  
4.5.7 Minimum processing time ....................................... 46  
4.5.8 Post processing .................................................... 46  
4.6 Introducing Fixed Productions, Maintenance Stops and Production Line Shut Downs .................................................... 49
4.7 Final Model Configuration .................................................. 50

5 Evaluation of Constraint Programming Model .................................. 51
  5.1 Search Limit and Solution Determinism .................................... 51
  5.2 Validation Cases ............................................................. 52
  5.3 Effect of Running Time Limit ............................................... 52
  5.4 Effect of Objective Weights ................................................ 54
  5.5 Alternative Objectives ...................................................... 56
    5.5.1 Alternative expression for non delivery and inventory deficit costs 56
    5.5.2 Alternative expression for setup costs ................................ 59
  5.6 Effect of Random Seed ..................................................... 60
  5.7 Constraint Relaxation ...................................................... 62
  5.8 Test on Larger Data Instance .............................................. 63
  5.9 Comparison of Model and Manual Solution ............................... 64

6 Final Improvements .............................................................. 67

7 Conclusion ........................................................................... 69
  7.1 Future Work ................................................................. 70

Bibliography ............................................................................. 76
Chapter 1

Introduction

As competition gets more fierce and customer demands rise, high quality planning and scheduling is of great importance for the process industry. Despite the considerable research and the progress made in the fields of planning and scheduling during the last decades [16], problems like the production planning and scheduling problem at AC continue to pose a great challenge. As defined in [16], production planning determines the operational plans that are used to structure the future production, distribution and other related activities according to the business objectives (like market penetration, top-line growth, etc.). In comparison, scheduling is more detailed as it is decided when exactly each activity will be executed and by which resource. In other words, planning and scheduling is about choosing which products to produce, when, and through which resources in the production facility. In such problems there are usually many contradicting objectives that must be taken into account, like cost minimization, minimum makespan, highest profit, reliability, customer satisfaction, etc. As a result, planning and scheduling problems tend to be very complex and are usually dealt by choosing approaches like Constraint Programming (CP), Mixed Integer Programming (MIP), and combinations thereof.

This graduation project is part of a larger research project by TU/e and AC and is a continuation of the MSc projects of Roel Coset [6] and Joost van Twist [29]. Initially an automated planning tool was developed in [6] using IBM ILOG Plant PowerOps (PPO) [14], which was further investigated and improved in [29]. The ultimate goal of the project is to develop a model that is as accurate as possible and that is considered useful (or has the potential to be so) by AC. Thus, the results obtained by [6, 29] are used as a starting point, but rather than working on further improving the already developed model on PPO, we concentrated on the development of a completely new model using IBM ILOG CPLEX Optimization Studio [13]. The reason for this decision is not only that IBM decided to take PPO off the market, but also because in [29] it was shown that with the particular method the results were not reproducible and the solution method had a high variability (non-determinism), to the point where the model was considered to not be useful. This meant that running the solution method twice for the same data instance, would give two different results and also the quality could have significant differences. To that extent, two new deterministic models, one using Mixed Integer Programming and one using Constraint Programming, are developed. Furthermore, in order to improve the performance of the CP model, a post processing method is also developed comprising of three steps. In addition, the input data and the plant constraints (as obtained by [29]), are revised and updated in an attempt to reflect better the current situation in the production facility of AC.

Regarding the MIP model that we developed, it is mainly based on the mathematical formulation proposed in [29]. By developing this model, we investigate the applicability and performance of the particular formulation to the problem of AC. Since this MIP model gives unsatisfactory results, a deterministic CP model is developed to investigate if
it can handle the particular problem better. Then, several alternative ways of modeling are analyzed for the CP model, to get an understanding of the effect that they have on the model performance. As a next step, multiple tests are performed to investigate the model’s behavior. This investigation includes tests on different run time limits and objective weights, alternative objective functions, as well as tests after relaxing particular constraints. Also a test on a bigger data instance (bigger than the ones used in [6] and [29]) is also performed, to investigate the development of the solution quality, when the model is used for a longer time period. Finally, a comparison is made between a model solution and the corresponding manually generated plan, to check the model solution quality on a deeper level.

In the remainder of this chapter short descriptions of CP and MIP are presented, along with the literature review that was performed regarding planning and scheduling. More specifically, Section 1.1 gives a description of Constraint Programming, while Section 1.2 presents Linear Programming and Mixed Integer Programming. A literature review regarding hybrid methods that combine MIP and CP to solve planning and scheduling problems is presented in Section 1.3, and finally, some alternative approaches that were found in the literature are presented in Section 1.4.

1.1 Constraint Programming

Constraint Programming (CP) is a problem-solving paradigm for combinatorial problems that has proven to be both efficient and effective in a wide range of application areas. A Constraint Satisfaction Problem (CSP) is a combinatorial problem comprising a set of variables $X = \{x_1, x_2, ..., x_n\}$, a set of domains $D = \{d(x_1), d(x_2), ..., d(x_n)\}$ such that all variables $x_i \in X$, $i \in [1, n]$, have a value in their respective domain $d(x_i)$, and a set of constraints $C = \{c_1, c_2, ..., c_k\}$ on $D$. In Constraint Programming, constraints are relations between the variables, specifying the properties of the solution to be found. A constraint $c_1 : d(x_1) \times d(x_2) \times ... \times d(x_n) \rightarrow \{true, false\}$ defines which combinations of values satisfy constraint $c_1$. The constraints used can be of various kinds, like “$x_1$ or $x_2$ is true” or “$x_3 \geq 0$”. The problem is then to find an assignment (also called solution) for all variables, such that all constraints $c_j \in C$, $j \in [1, k]$ are satisfied. An important characteristic of CP is the clear distinction between the problem statement and the strategy that is followed for solving the problem. More specifically, the programmer defines the problem by declaring the variables and the various constraints that have to apply for an acceptable solution. Additionally, although it becomes less and less true, the programmer can define the search heuristics that will be used for finding the solution of the problem, as an alternative to the predefined heuristics that are offered. CP has proven to be an important problem-solving tool for scheduling problems, leading to the development of the field of Constraint-Based Scheduling [1, 9, 22].

According to [22], the central concept of CP is constraint propagation. In the process of constraint propagation, a subset of constraints and domains is used to produce more restrictive constraints or domains. In other words, the domains of all dependent variables are recomputed each time the current domain of a variable is changed. This iterative procedure is terminated when no more changes occur. What is most important with constraint propagation, is that it simplifies the constraint problem to a large extent and thus increases the efficiency of the search algorithm. In practice, the most important local consistency property that is used to justify the produced constraints/domains is arc-consistency. As such, arc-consistency has received a lot of attention from the literature. In a given CSP, if all constraints are binary (i.e. they affect two variables), a constraint graph can be constructed, using nodes to represent the variables and edges between nodes to represent the constraints. A directed arc $(x, y)$ is arc consistent, if for each value in the
domain of \( x \), there is a value in the domain of \( y \) that satisfies the constraint \( C_{x,y} \) between \( x \) and \( y \), and the other way around. If this is not true for a value in the domain of \( x \), then this value can be safely removed. By removing all such values, the arc \((x,y)\) becomes arc consistent \([3]\). Figure 1.1 gives an arc consistency example of constraint \( C_{x,y} \) \((x < y + 1)\) between variables \( x \) and \( y \). In (a) the original domains are shown, while in (b) the domains of \( x, y \) have been reduced to make \((x, y)\) arc consistent. In [21], a filtering algorithm is proposed to implement arc consistency for global cardinality constraints efficiently, while the authors of [2] propose a general schema (named GAC schema), that implements arc-consistency on non-binary constraints.

\[
\begin{array}{cccc}
\text{x} & \in & \{3, 4, 5, 6\} & \text{x} \leq \text{y} + 1 & \text{y} & \in & \{1, 2, 3, 4\}
\end{array}
\]

(a) Original domains

\[
\begin{array}{cccc}
\text{x} & \in & \{3, 4\} & \text{x} \leq \text{y} + 1 & \text{y} & \in & \{3, 4\}
\end{array}
\]

(b) Domains for \((x,y)\) to be arc consistent

Figure 1.1: Creation of arc consistency: In (a) the original domains are shown, while in (b) the domains are adjusted to make \( x < y + 1 \) arc consistent.

CSPs are mainly solved using either backtracking search or a combination of backtracking and Local Search (LS) \([22]\). The backtracking algorithm performs a depth-first traversal of a search tree, using the constraints to eliminate any subtrees that do not contain any solution. With backtracking it can be shown whether a problem has a solution or not. Furthermore, in case there is a solution for a specific CSP, backtracking guarantees to find the optimal one. On the other hand, local search algorithms may be effective in finding an approximation of the optimal solution (if any), but they cannot show if a problem does not have a solution at all \([22]\).

As shown in \([17]\) in their most general form, CSPs are NP-hard problems. Thus it is not possible to find efficient general-purpose algorithms for solving all types of constraint problems. In \([22]\) it is analyzed how the types of constraints affect the complexity of solving a CSP. Finally \([22]\) also describes the problem of symmetry and presents the three main ways to deal with it:

- Problem reformulation.
- Adding symmetry breaking constraints.
- Ignoring symmetric states using search strategy modifications.

### 1.2 Linear Programming and Mixed Integer Programming

Linear Programming (LP) is used for solving problems of minimizing or maximizing a linear relation that is subject to certain linear inequalities. Let \( A, b, c, x \) be a matrix and three vectors respectively. Then two basic LP examples are the following \([24]\):

\[
\max \{ cx | Ax \leq b \} \text{ and } \min \{ cx | x \geq 0; Ax \leq b \}
\]
Any vector \( x \) satisfying the equation \( Ax \leq b \) is a feasible solution and \( cx \) is called the objective value of \( x \). The difference between Integer Linear Programming (ILP) and LP is that in ILP the variables can only be integers. A form of ILP problems is the following [24]:

Let \( A, b, c \) be a rational matrix and two rational vectors respectively. The problem is to find the solution for \( \max \{ cx \mid Ax \leq b; x \text{ integral} \} \).

An alternative form is to find the solution for \( \max \{ cx \mid x \geq 0; Ax = b; x \text{ integral} \} \). ILP problems are NP-complete and this is in accordance with the general practical experience that ILP problems are hard to solve and time consuming when using present-day methods. The proof for the NP-completeness of ILP problems can be found in [24]. Finally, [24] gives a definition for the LP-relaxation of ILP problems. More specifically, the LP-relaxation of \( \max \{ cx \mid Ax \leq b; x \text{ integral} \} \), is the LP-problem \( \max \{ cx \mid Ax \leq b \} \) which obviously, gives an upper bound for the corresponding ILP problem.

When only some of the variables are restricted to be integers, then we have a Mixed Integer Programming (MIP) problem. MIP is used abundantly for solving combinatorial optimization problems like production planning problems, vehicle routing, etc. A common method to solve MIP models is by way of branch and bound. Branch and bound is basically a “divide and conquer” strategy, where the feasible region (i.e. the acceptable values of decision variables) is divided in smaller, more manageable regions. For instance a variable \( x \) with range \( k_1 \leq x \leq k_2 \), can be divided into two subproblems with ranges from \( k_1 \) to \( m \) and from \( m + 1 \) to \( k_2 \) respectively [8]. After dividing the problem into subproblems, each subproblem is solved using LP-relaxation. After calculating the LP-relaxation of a subproblem, if either of the following is true:

- the subproblem has no feasible solution,
- the optimal solution of the LP relaxation is integer (i.e. all unknowns get integer values), or
- the optimal solution of the LP relaxation is worse than the best known solution so far,

then the specific subproblem does not need to be divided any more, as we know everything that we need about it. In this case the subproblem is called fathomed [25]. However, if the optimal solution of the subproblem found by the LP is better than the best known, but it is not integer, then the specific subproblem must be divided again. In case the optimal solution of a subproblem is integer, this solution must be compared with the best known so far, as it can be the new best solution. Finally, if all subproblems have been solved without finding a feasible solution, the algorithm terminates and the problem is considered infeasible.

An alternative to branch and bound is the cutting-plane algorithm. Instead of dividing the main problem into subproblems, the cutting-plane algorithm works with a single LP problem, which is continuously refined by adding cuts. Cuts are constraints that are added to an LP problem until the optimal basic feasible solution takes integer values [28]. As defined above, a feasible solution (also called basic feasible solution [23]) for the LP \( \min \{ cx \mid Ax \leq b \} \) is any vector \( x \) that satisfies the equation \( Ax \leq b \), and according to [23], the feasible region \( U \) of the LP is the set of such feasible solutions. If \( U = \{ x^1, x^2, ..., x^k \} \), then the optimal basic feasible solution is vector \( x^i \), \( i \in [1, k] \), if and only if \( cx^i \leq cx^j \), \( \forall j \in [1, k] \) where \( j \neq i \). Back to [28], a cut for a specific fractional solution (i.e. a solution where some variables have non-integer values), has to be satisfiable by all feasible integer solutions. On the other hand, the current fractional solution must be infeasible for the cut. An example of a cut is shown in Figure 1.2. In this graph, all feasible solutions of the
problem are marked with red dots in the area to the left and bottom of each constraint (blue dashed lines), while the fractional solution is the point where the dashed line of constraint 1 intersects the dashed line of constraint 2. After adding the cut (black dashed line), all feasible solutions that are satisfied by both constraints are to the left of the black dashed line. On the contrary the fractional solution is to the right.

Figure 1.2: A cut (black dashed line) is used to leave the fractional solution (black circle) out.

1.3 Mixed Integer Programming and Constraint Programming Integration

In contrast to Sections 1.1 and 1.2 that described Constraint Programming (CP) and Mixed Integer Programming (MIP) in general, this section is about solving planning and scheduling problems, by combining the strengths of both MIP and CP based methods. According to [5], several problems have been used by many authors to compare MIP and CP based methods. For instance, the findings of [9] and [11] demonstrate that MIP methods are efficient when the objective functions have many variables. On the other hand, CP based techniques seem to perform better than MIP in highly constrained discrete optimization problems that have only a few variables in the objective function or when searching for a feasible solution.

From the above, it is deduced that if the two techniques are combined, their individual strengths and advantages can also be combined and help to solve problems that would otherwise be impossible or very hard to solve [5, 15]. Two alternatives for combining MIP and CP methods are either by forming a hybrid model consisting of constraints from both MIP and CP, or by decomposing the problem into two subproblems. In the latter case, the first subproblem is solved using MIP, while the second subproblem is solved with CP [10, 11, 12]. In the rest of the section, some decomposition method examples are presented.

A MIP/CP decomposition method is proposed in [15] that is similar to the Generalized Benders Decomposition [7]. Let $x \in X$, $y \in Y$ be vectors of variables and $G$ be a vector of constraint functions for $x$ and $y$. Then for a problem of the form:

$$\max \{ f(x, y) \mid G(x, y) \geq 0 \}$$

where vector $y$ contains complicating variables (i.e., variables that when they are temporarily fixed, the remaining of the optimization problem becomes considerably more tractable), there are substantial opportunities for reducing the problem complexity by considering the problem in the $y$-space rather than in the $xy$-space. Benders Decomposition is based on
the above and instead of solving a large optimization problem by considering all decision variables and constraints, the method divides the problem into multiple smaller problems. That way, solving these smaller problems iteratively becomes more efficient than solving the initial larger problem. Back to [15], the MIP/CP model consists of separate MIP and CP constraints. Furthermore some relations are used to express the equivalence between MIP and CP variables. At the beginning, a relaxed MIP model is solved to optimality. Unless there is no solution - meaning that the model is infeasible - the equivalence relations determine the values of the CP variables and then the CP feasibility problem is solved, trying to extend the partial solution obtained by the MIP model, to a complete solution that satisfies all constraints. If the CP problem leads to a feasible solution, then this solution is the optimal one and the whole process is terminated. The solution of the CP problem is optimal for the main problem, as it belongs to its domain and the objective value equals the optimal objective value found by the MIP model, which is a relaxation of the main problem. If however there is no feasible solution, the causes for infeasibility are inferred and according to them, new cuts are added to the MIP model. Then the MIP problem is run again. The entire procedure is repeated until either the MIP model becomes infeasible or a feasible CP solution is obtained. It is important to note that since these cuts may be weak, problem-specific cuts should be used when this is possible. These stronger cuts exploit the structure of the specific problem and they do not only remove the partial solutions that caused the infeasibility, but also any other partial solution that has similar characteristics. [15] proposes two ways to implement the above given method. The straightforward way is to solve the MIP model from the beginning, each time new cuts are introduced. The second way, which is more efficient, is to use the branch-and-bound tree from the previous run of the relaxed MIP model, and update it when new cuts are added (for an example, see [20]).

In [19] a MIP/CP decomposition algorithm is proposed, for solving short-term scheduling problems of batch plants that rely on the State Task Network (STN) representation. As explained in [29], STNs are directed graphs containing State and Task nodes. State nodes are used to represent raw materials as well as intermediate and final products. As to Task nodes, they represent processing tasks and they are related to State nodes to represent the amount of materials or products that are consumed or produced. The algorithm of [19] is based on a MIP model, proposed earlier in [18] by the same authors, and it consists of two phases. The first phase is a preprocessing that determines the earliest start times (EST) and latest finish times (LFT) of tasks and units. Furthermore, in this phase strong integer cuts are determined and added in the cut-pool of the master problem of the second phase. Then the second phase follows, consisting of an iteration between the master problem - solved with MIP - and the CP subproblem. MIP is used for the optimization of the high level decisions (e.g.: size of batches, assignment of tasks to units, etc.), while CP is used for determining a feasible detailed schedule. After each iteration, new integer cuts are added to the problem to exclude any infeasible or previously obtained assignments. When the upper bound of the MIP objective function gets less than or equal to the lower bound of the CP objective function, the algorithm terminates.

1.4 Alternative Approaches

There are several alternative approaches for solving scheduling problems in the (batch) process industry, than the ones presented above. One such alternative approach is proposed in [26], where the problem is separated in a planning and a scheduling problem. The planning problem is used to determine the size of batches and the number of executions of each process in order to satisfy the requirements for the products. On the other hand, scheduling determines the exact schedule of processing units and storage facilities
over time, for processing the corresponding batches. Regarding the planning problem, a novel formulation is presented that uses MIP. According to the proposed method, the planning problem consists of defining the batch size and the number of executions of each task that are needed to satisfy the requirements. Additionally, the specified batch sizes are observed and the total bottleneck workload is minimized, while it is made sure that the perishable products (if any) are consumed immediately after production. As to the scheduling problem, it is solved with a priority-rule based scheduling method, where the start time of each task execution is determined so that it does not violate any constraints.

An interesting aspect taken into account by [26], is the limitation of the storage capacity. For this, the authors introduce the concept of unscheduling step. To deal with any capacity overflows of intermediate products at a particular time $t$, an eligible operation $j$ is scheduled before $t$, that will consume the specific product. However, if this is not possible, an unscheduling step is performed in the following way. Let $t_j$ be the earliest feasible start time of operation $j$, and let $i$ be the operation that causes the capacity overflow. Also let $r_i$, $p_i$ denote the earliest start time and required processing time of $i$. Then if $r_i$ is postponed by $t_j - p_i$, it is ensured that $i$ will be finished at least at time $e_i \geq t_j$ (since $r_i = t_j - p_i$ is the earliest start time of $i$, and $i$ requires $p_i$ time to complete). Thus it will be possible for operation $j$ to start at time $e_i$ (or earlier) and consume the amount that cannot be stocked because of the overflow. If however a maximum number of such steps is performed without finding a feasible solution, the method is terminated. The detailed description of the unscheduling steps is given in [26]. Finally, any idle times that are introduced by the unscheduling steps, can be easily removed by a post-processing step.

In [27], written by two of the authors of [26], a cyclic sub-schedule is computed and it is then applied several times to satisfy the primary requirements for a set of final products. More specifically, this approach consists of three phases, each of which is performed only once. The first phase is called cyclic batching phase, which determines the set of operations performed in one cycle along with the respective batch sizes and the input and output proportions (i.e., the amount of products that are put in and out during each cycle). Then, the number of cycles that are needed is determined. For this phase, it is important to limit the number of operations per cycle, or the scheduling problem may become intractable. Furthermore, the quantity of intermediate products produced in each cycle must be equal to the amount consumed, or it will not be possible to apply the acquired sub-schedule solution of the next phase iteratively.

During the second phase - called cyclic batch-scheduling - the processing units, intermediate products and storage facilities are allocated over time to the cycle operations. While the schedule is computed, the unscheduling steps introduced in [26] can be applied to overcome any deadlocks that may occur. If the maximum number of such steps is performed, then the whole procedure is terminated and the problem is considered infeasible. As in [26], post-processing can be performed to remove any idle times introduced by the unscheduling steps. Then the last phase follows, which is the concatenation. In that phase, the required number of identical sub-schedules is connected to each other, to give the complete scheduling solution.

The authors of [27] conclude that the first phase is independent of the primary requirements and consequently independent of the number of operations. Furthermore, since the size of the cycle can be reduced explicitly, only the third step is affected by the primary requirements. However, the concatenation step is performed very efficiently. So, by using the cyclic approach, an NP-hard scheduling problem is shifted to a polynomially solvable concatenation problem.

The last approach that we discuss here is from the IBM ILOG Plant PowerOps (PPO) package that was used in both [6] and [29]. PPO performs optimization within three separate modules, which are planning, batching and scheduling. The planning engine of
PPO makes use of MIP solved with IBM ILOG CPLEX. The planning module generates rough production plans by considering the various constraints of the production lines. The production timeline is divided into buckets and PPO defines the amount of each product that is to be produced in each production line during each bucket. After finishing the planning, the batching module takes over that constitutes a link between the planning and scheduling engines, using MIP, CP and heuristics. Given the results produced during planning, batching creates the overall flow from raw materials to final products that satisfies the customer demands. Finally, the scheduling module makes use of CP based on IBM ILOG Constraint Programming. During this phase detailed schedules are created with production orders having exact start and end times. Complex constraints (e.g.: resource compatibility, changeovers, cleanups, etc.) are taken into account during scheduling and also rescheduling capabilities are offered, allowing for instance the user to manually edit the produced schedule [14].

The remainder of this thesis is organized as follows. Chapter 2 gives a general description of the problem and also presents various aspects and limitations (constraints) of the production facility of AC. The mathematical formulation that is used for the implementation of the MIP model is presented in Chapter 3. The same chapter presents the performance evaluation of the model, as well as the various modifications that were tested while trying to improve the model. Next, Chapter 4 presents the CP model that was developed. Again, several improvement steps as well as other additions to the CP model are presented in that chapter. However, the evaluation of the CP model and its results on various tests are given in Chapter 5. Then, some last modifications for improving the CP model are tested in Chapter 6. Finally, the conclusion of the report is given in Chapter 7, along with several suggestions for future work.
Chapter 2

Problem Description

In this chapter, Section 2.1 gives a brief specification of the planning and scheduling problem at AC. Next, the required changeovers and the constraints of the plant are presented in Sections 2.2 and 2.3 respectively. Finally the problem input is described in Section 2.4, after which, Section 2.5 gives the definition of the AC Planning and Scheduling Problem.

2.1 The Planning and Scheduling Problem

As described in Chapter 1, the main planning and scheduling problem for AC is to determine what products will be produced on the production lines and at which time, while respecting the various technical constraints of the plant. Besides the constraints, an objective is also provided that needs to be minimized or maximized in order to get an optimal solution. In the problem of AC, the most important objectives are the satisfaction of as many production demands as possible, while preserving the targeted inventory levels of each product by the end of each month. Each production line can produce a single product at a time, according to a variety of recipes that are possible on each line, which also determine the production rate of the corresponding product. Finally, when a production line needs to change recipe, a changeover process intervenes during which the production line does not produce any product. As it is stated in [29], the production process at AC is a continuous process, since all different steps happen continuously and they do not require a separate planning.

2.2 Changeovers

When a production line changes the executed recipe, a changeover process is performed and the production on the production line is suspended. Obviously, its duration depends on the different parameters between the two corresponding products. In total there are four different changeover types (i.e.: soft, normal, blocking and die change), each of which has a different duration. However, since most of the information regarding the changeovers is considered classified, more details are not presented in this version of the document.

2.3 Technical Constraints

There are several technical constraints that must be considered by the model. However, they are classified and they are not included in this document version. Besides the technical constraints, there are several other (non-technical) constraints that must be considered, since they restrict the planning solution even further. These are the following:

1. Production of products $P_1$, $P_2$, and $P_3$ can only start during weekdays.
2. Changeovers of type *die*, which have the largest changeover duration, can only occur during weekdays.

3. For any production, the minimum production duration is 5 days. In practice, this can be 5 hours more or less and the exact time depends on the recipe that was running previously on the same production line. Because of difficult modeling issues though, a minimum period of 5 days is assumed for all cases.

4. For some types of product, the suitable production lines are further limited, due to a difference in unit size, or other commercial requirement. For instance, a commercial requirement is that some products have to obtain some chemical properties and this can only be made on specific production lines. To model this, different alternatives are created for these products and each of them can be produced only on some particular lines.

### 2.4 Problem Input

The input of the scheduling problem comprises many elements that are presented in this section. These elements are used as input by the MIP and CP models presented in Chapters 3 and 4 respectively, and along with the constraints given in Section 2.3 they define the problem for any particular data instance.

*Time* is one of the main concepts of the model. The time unit is hours, as demand deadlines, production times, changeover durations, etc., are all expressed in hours. As to the length of the planning horizon of a specific instance, this is denoted by $H$.

There is a wide range of *products* that are produced in the plant and $P$ is the set of different products at an aggregated level. A single $p \in P$ represents an aggregation of a number of products at SAP’s material level. A product at SAP’s material level is a packaging of units, with a specific unit size, that contains a specific weight of a specific product, made with a specific technology. AC maintains a forecast at this product level. For many customers, characteristics like packaging, unit size or unit weight do not matter. As a result, the forecasts for some different products at the material level, are forecasts for the same product at the aggregated level.

For every product $p \in P$ there are several *production orders* that express monthly forecasts of the demand for a specific product. The set of production orders is denoted by $O$, and each order $o \in O$ for product $p_r \in P$ has a production deadline $due_o$ for a quantity $dem_o$. An important measure for production orders is the *non-delivery cost* ($ndc_o$) per quantity unit (1 ton), which is defined as the contribution margin of product $p_r$ ($contribution_{p_r}$) multiplied by a modifier ($ndcModifier_o$) that is specific for each production order and emphasizes its strategic importance. $ndc_o$ is thus given by the following formula:

$$ndc_o = contribution_{p_r} \times ndcModifier_o \quad (2.1)$$

Furthermore, each product can be produced by one or more *recipes* belonging to the set $R$ of recipes.
Another important concept that is given as input to the mathematical model is the set of **resources**. Resources are divided in primary and secondary resources and they are used during the production of products, according to the needs of the respective recipe. When the production of a recipe finishes, all used resources are released and are available for other recipes. In some cases though, a *cleaning* process has to precede before a specific resource is ready to be used again. The difference between primary and secondary resources is that the primary resources actually produce the product, while the secondary resources constrain the primary resources with a specific capacity. The various primary resources are presented below:

- The set of high-rise resources is denoted by $HR$ and each $h \in HR$ has a specific throughput capacity $hrCap_h$.
- The set of production lines is denoted by $L$ and every $l \in L$ is linked to a specific high-rise resource.
- Finally, the set of dies is denoted by $SP = \{S, D, E, F, K, P, X, L\}$. For each die $s \in SP$, there is a given capacity $spCap_s$ and each recipe $r \in R$ requires a certain type of die $die_r$ during the whole duration of its production.

Between every pair of different recipes $r, r' \in R$ that are produced in sequence on the same production line, a *changeover* takes place. Obviously, the duration of a changeover depends on the changeover type, which has four alternatives: spin, block, soft, and normal. As explained, the duration of changeovers is expressed in hours and is given by $ct(r, r')$. If $klit_r$ denotes whether recipe $r$ needs klit-machinery or not and $unitweight_r$, $polymer_r$, and $die_r$ denote the weight of the unit, the polymer concentration, and the die needed by recipe $r$ respectively, then $ct(r, r')$ is given by the following equation:

$$
ct(r, r') = \begin{cases} 
0 & \text{if } r = r' \text{ — no changeover}. \\
0 & \text{if } (klit_r \neq klit_{r'} \lor unitweight_r \neq unitweight_{r'}) \land \\
\text{polymer}_r = \text{polymer}_{r'} \land die_r = die_{r'} \text{ — soft changeover}. \\
7 & \text{if } |\text{polymer}_r - \text{polymer}_{r'}| \geq \text{polymer}_{\text{block}} \text{ — block changeover}. \\
24 & \text{if } die_r \neq die_{r'} \text{ — die changeover.} \\
3 & \text{otherwise — normal changeover.}
\end{cases}
$$

In the above equation, if more than one condition is valid for a pair of recipes, then the largest value applies.

For each production line, a recipe is given that corresponds to the recipe that was executed on the specific line, just before the beginning of the new scheduling horizon. By providing these *current productions* to the model, the respective changeover tasks can be scheduled just before scheduling new recipes.

All possible combinations of products, recipes, and production lines form the set of **spinning tasks** $I^{spin}$. Whether a combination is valid and is included in $I^{spin}$ or not, depends on various constraints. However, since these constraints are classified, they are not included in this document.

$$I^{spin} = \{(p, r, l) \in P \times R \times L \mid \text{all classified technical constraints between } p, r, l \text{ are satisfied}\}$$
Besides the spinning tasks, there are changeover tasks as well. The set of changeover tasks \( I^{co} \) consists of all possible combinations of two recipes \( r, r' \in R \) and a production line \( l \in L \). The condition for such a combination to be included in \( I^{co} \), is that \( ct(r, r') \neq 0 \) and also that \( l \) is able to produce both \( r \) and \( r' \).

\[
I^{co} = \{(r, r', l) \in R \times R \times L \mid ct(r, r') > 0 \land l \in lne_r \land l \in lne_{r'}\}
\]

Finally, for each spinning task it is important to specify the required amount \( \mu \) of capacity of each resource. In all following cases, \( i \in I^{spin} \) and depending on the kind of resource, different index symbols are used for \( \mu \):

- High-rise resources \((h \in HR)\):
  \[
  \mu_{hr}^i = \begin{cases} 
  \text{tp}_{r(i)} & \text{if } hr_{Res(i)} = h \\
  0 & \text{otherwise}
  \end{cases}
  \]

- Production line resources \((l \in L \text{ and } i' \in I^{co})\):
  \[
  \mu_{l}^i = \begin{cases} 
  1 & \text{if } l(i) = l \\
  0 & \text{otherwise}
  \end{cases}
  \]

- Evaporation resource:
  \[
  \mu_{ar}^i = ar_{r(i)}
  \]

- Die resources \((s \in SP)\):
  \[
  \mu_{sp}^i = \begin{cases} 
  1 & \text{if } die_{r(i)} = s \land l(i) \notin \{PL_4, PL_5, PL_8, PL_9\} \\
  0.5 & \text{if } die_{r(i)} = s \land l(i) \in \{PL_4, PL_5, PL_8, PL_9\} \\
  0 & \text{otherwise}
  \end{cases}
  \]

- Dica resource:
  \[
  \mu_{dica}^i = \begin{cases} 
  1 & \text{if } dica_{r(i)} \leq 10 \\
  0 & \text{otherwise}
  \end{cases}
  \]

Note in \( \mu_{sp}^i \) that production lines \( PL_4, PL_5, PL_8, \) and \( PL_9 \) use half a die, while the rest of the production lines use a whole.

### 2.5 Problem Definition

This section gives the definition for the planning and scheduling problem at AC.

**Definition:** AC Planning and Scheduling Problem (TASP).

An instance of TASP consists of the input data as described in Section 2.4. The most important components are the set of spinning tasks \( I^{spin} \) and the various sets of resources \((HR, L, SP)\). A solution of TASP is defined as the allocation of tasks within the planning horizon \( H \). The problem is then to find a solution that respects all constraints as defined in Section 2.3, while minimizing the objective function \( OF \) as defined in Section 3.4. Finally, different solutions of the TASP are comparable with one another only by comparing the solution total costs as given by \( GO \) in Section 3.4.
Chapter 3

Mixed Integer Programming Model

This chapter gives the description of the mathematical formulation that is used for the implementation of the MIP model. This model uses the model described in [29], which in turn was based on [4], as a starting point. In [4] a general mathematical formulation is proposed for calculating the schedule of batch and/or continuous multipurpose plants. This formulation is based on the Resource-Task Network (RTN) representation, that uses two types of nodes (i.e. resources and tasks) to represent processes as sets of topological entities. It is important to note that the model presented here is very similar to the one proposed in [29]. However, during the implementation of the MIP model we found that many concepts could not be modeled in practice and needed to be reformulated or adjusted accordingly. Furthermore, some new equations have been added to model constraints that were not considered before.

The rest of the chapter has the following structure. Section 3.1 gives a description of the input needed by the mathematical model, while Section 3.2 describes the model output. Next, Section 3.3 presents the mathematical formulation of the various constraints that apply in the plant. Section 3.4 gives the description of the objective function that is used for the optimization of the output and then, Section 3.5 presents various alternative ways that were used to model some particular constraints, in an attempt to improve the model performance. Finally Section 3.6 presents the testing that was performed to verify the model’s correctness and to evaluate its performance.

3.1 Input

The problem input as already defined in Section 2.4, is how it is used by the MIP model. There is only a small addition regarding time and it is presented below.

As said, the time unit is hours and \( H \) denotes the length of the planning horizon. In the MIP model, \( H \) is spanned by a series of event points \( T = \{1, 2, ..., T_{\text{max}}\} \), where \( T_{\text{max}} \) is the maximum number of such event points. In general, as \( T_{\text{max}} \) grows larger, the accuracy of the model becomes better, leading to better solutions. However, as \( T_{\text{max}} \) increases, the computational effort for solving the model increases as well. Thus, a balance has to be found that will lead to good solutions within acceptable computation times. Another important variable is \( \Delta T \) that denotes the maximum number of time points that any spinning task (described later) can overlap. Just like \( T_{\text{max}} \), the value of \( \Delta T \) defines a trade-off between model accuracy and computational effort.
3.2 Output

Given the aforementioned input data, a model solution is the allocation of tasks (either $I^{\text{spin}}$ or $I^{\text{co}}$) to pairs of event points. As explained, $\Delta T$ denotes the maximum number of time points that any spinning task can overlap. The allocation of a spinning task $i \in I^{\text{spin}}$ to two event points $t, t' \in T$, where $t < t' \leq t + \Delta T$ is denoted by the binary variable $N_{i,t,t'}$, which is 1 if $i$ runs from $t$ until $t'$. Similarly, a changeover task $i \in I^{\text{co}}$ can only run between two consecutive time points. So, for $t \in T$, where $t < T_{\text{max}}$, the binary variable $N_{\text{co},i,t,t+1}$ is used. During the time period that a task (either spinning or changeover) is running, the required resources are used according to the corresponding $\mu$ variables and they are released exactly at the time point when the task finishes execution. When a spinning task runs between two event points, an amount $\xi_{i,t,t'}$ of the respective product is produced. Another important variable is $T_t$, which represents the absolute time corresponding to event point $t$. The produced amount $\xi_{i,t,t'}$ is given by the following equation:

$$\xi_{i,t,t'} = tph_r \times (T_{t'} - T_t),$$

where $tph_r$ is the amount of product produced by $r \in R$ (tons/h).

Furthermore, variable $I_{p,t}$ represents the inventory level of product $p \in P$ at event point $t$ and slack variable $psl_o,t$ denotes the non-delivered amount of production order $o \in O$ at event point $t$. Finally, another binary variable denotes whether production order $o$ is (partially) delivered at event point $t$. This variable is $po_{o,t}$ and it is important to note that for every production order $o$, $po_{o,t}$ can be 1 at a single event point only. In other words, it is not allowed to deliver a specific order in multiple portions. So, when $po_{o,t}$ is 1, the whole amount of product $p \in P$ that is delivered to (partially) satisfy order $o$ must be subtracted from the inventory level $I_{p,t}$.

![Figure 3.1: Model output example for three production lines [29].](image)

An example of the output, originally presented in [29], is given in Figure 3.1. As it can be seen in the figure, task $i_2 \in I^{\text{spin}}$ runs from event point 5 until event point 9. This means that $N_{i_2,5,9} = 1$, while no other task can run during this time period on the same production line. Thus, for all $i \in I^{\text{spin}}$ where $i \neq i_2$ and $l(i) = l(i_2)$, $N_{i,5,9} = 0$. Also $N_{i,t,t'} = 0$ for all $t,t' \in T$, $t < t' \leq t + \Delta T$, that have time overlap with time period $[T_5, T_9]$ (i.e., $t \in \{6, 7, 8\}$ $\vee$ $t' \in \{6, 7, 8\}$). As to the amount produced by $i_2$, it is given by:

$$\xi_{i_2,5,9} = tph_{r(i_2)} \times (T_9 - T_5)$$

Where $r(i_2)$ is the recipe $r \in R$ of task $i_2$. Finally $\Delta T = 4$, meaning that there is no task overlapping more than $\Delta T - 1 = 3$ event points.


3.3 Constraints

This section presents the constraints of the model, separated in different categories according to their purpose.

**Timing constraints:** Timing constraints are the core of all continuous-time formulations [4], having a big influence on the model performance. Their purpose is to specify the amount of time that must pass between event points of tasks. Both Equations 3.1 and 3.2 are stated per production line and two different cases are distinguished. One case is for consecutive time points $t, t' \in T$, $t' = t + 1$, where changeover tasks are possible to be scheduled, while the other case is when $t' > t + 1$. In both equations the summation is allowed because during a specific time interval on a particular production line, only one task (either spinning or changeover) can be executed. This is ensured by the resource balance constraint given by Equations 3.9 and 3.10. Equation 3.1 states that when a changeover or spinning task is executed, the duration of the interval must be at least equal to the changeover time or the required production time respectively. The required production time is equal to the produced amount $\xi_{i,t,t'}$, divided by the production rate $tph_{r(i)}$ (denoting tons/h) of the recipe used. Equation 3.2 is used to specify an upper bound on the duration of the interval, setting it at most equal to the required changeover time (for changeover tasks) or at most equal to the production time needed (for spinning tasks). Additionally this equation states that if no task is run during an interval, then its duration can be up to the length of the planning horizon $H$.

For all $l \in L, t, t' \in T \mid t < t' \leq t + \Delta T, t < T_{max}$

If $t' = t + 1$ :

$$
T_{t'} - T_t \geq \sum_{i \in I^{co}} N_{i,t,t'}^{co} \cdot cht_i + \sum_{i \in I^{spin}} \frac{\xi_{i,t,t'}}{tph_{r(i)}}
$$

If $t' > t + 1$ :

$$
T_{t'} - T_t \geq \sum_{i \in I^{spin}} \frac{\xi_{i,t,t'}}{tph_{r(i)}}
$$

If $t' = t + 1$ :

$$
T_{t'} - T_t \leq H \left( 1 - \sum_{i \in I^{co}} \frac{N_{i,t,t'}}{l(i)=l} - \sum_{i \in I^{spin}} \frac{N_{i,t,t'}}{l(i)=l} \right)
+ \sum_{i \in I^{co}} \frac{N_{i,t,t'}^{co} \cdot cht_i + \xi_{i,t,t'}}{l(i)=l} + \sum_{i \in I^{spin}} \frac{\xi_{i,t,t'}}{tph_{r(i)}}
$$

If $t' > t + 1$ :

$$
T_{t'} - T_t \leq H \left( 1 - \sum_{i \in I^{spin}} \frac{N_{i,t,t'}}{l(i)=l} \right) + \sum_{i \in I^{spin}} \frac{\xi_{i,t,t'}}{tph_{r(i)}}
$$

Besides the constraints of Equations 3.1 and 3.2, there are some other constraints relative to time, described by Equations 3.3 – 3.5. Equation 3.3 forbids the production of
some specific products (contained in the weekPr set) from starting during the weekends (non-technical Constraint 1). The start time of every weekend in the planning horizon is contained in the set Twe and they are expressed as the offset (in hours) since the beginning of the planning horizon. So, for every w ∈ Twe and every pair of event points t, t' ∈ T where t < t' ≤ t + ∆T, no task corresponding to a product p ∈ weekPr can occur if the time period t, t' overlaps with the weekend starting at w and finishing at w + 48. Unfortunately, Equation 3.3 is not in linear form. However, CPLEX has an automatic way of translating non linear equations to linear, at an expense of performance.

For all w ∈ Twe, i ∈ Ispin | pr_r(i) ∈ weekPr

\[ \sum_{t,t' \in T} N_{i,t,t'} \leq 0 \] (3.3)

Equation 3.4 ensures that every pair of different event points t, t' ∈ T represents different actual dates and also that if t < t', then T_t < T_{t'}.

For all t, t' ∈ T | t < t'

\[ T_t < T_{t'} \] (3.4)

Equation 3.5 is used for the objective function of the stock target (3.27), stating that there must be one event point at the end of each month of the planning horizon. This is needed because by the end of each month, it is checked whether the inventory levels are below the stock targets or not. The set T_month contains the offsets for the beginning of all months contained in the planning horizon, since the beginning of the planning horizon. For each element of this set, there must be exactly one event point whose actual date value is equal to that element. Again, this equation is not in linear form and CPLEX has to translate it itself, resulting in reduced performance.

For all m ∈ T_month

\[ \sum_{t \in T} (T_t = m) = 1 \] (3.5)

**Operational:** When a spinning task is executed, the produced amount must be limited within a minimum and a maximum value. The minimum value is equal to the minimum allowed production time mpt_r(i), of the recipe r(i) used by i ∈ Ispin, multiplied by the production rate of the recipe. On the other hand, since there is no upper limit for the production time, the maximum value is equal to the amount that would be produced, if the whole planning horizon was used.

For all t, t' ∈ T, i ∈ Ispin | t < t' ≤ t + ∆T, t' ≤ T_{max}

\[ mpt_r(i) \cdot tph_{r(i)} \cdot N_{i,t,t'} \leq \xi_{i,t,t'} \leq H \cdot tph_{r(i)} \cdot N_{i,t,t'} \] (3.6)

Note that mpt_r(i) must be larger than zero, or N_{i,t,t'} can be 1 while no material is actually produced.

**Resource balance:** The available quantity of a resource at any event point cannot be smaller than zero. This amount is equal to the capacity of the resource minus any amount that is required by tasks at the specific event point. Alternatively, the available amount of a resource at a given event point is equal to the available amount at the previous event
point, plus the amount that is released by tasks (either spinning or changeover) finishing at the current event point, minus the amount that is required by tasks starting at the current event point. For each type of resource, the corresponding equation is given below. Note in Equation 3.11 that lines $PL_4$, $PL_5$, $PL_8$, and $PL_9$ use half a die, while the rest of the production lines use a whole.

For all $h \in HR$, $t \in T$

If $t = 1$:

$$E_{hr}^{h} = hr \cdot cap_h - \sum_{i \in I_{spin}} \sum_{t' \in T, t < t' \leq t + \Delta T} \mu_{i,h}^{hr} \cdot N_{i,t,t'}$$

If $1 < t < T_{max}$:

$$E_{hr}^{h} = E_{hr}^{h-1} + \sum_{i \in I_{spin}} \left( \sum_{t' \in T, t-\Delta T \leq t' < t} \mu_{i,h}^{hr} \cdot N_{i,t,t'} - \sum_{t' \in T, t < t' \leq t + \Delta T} \mu_{i,h}^{hr} \cdot N_{i,t,t'} \right)$$

If $t = T_{max}$:

$$E_{hr}^{h} = E_{hr}^{h-1} + \sum_{i \in I_{spin}} \mu_{i,h}^{hr} \cdot N_{i,t,t'}$$

$$E_{hr}^{h} \geq 0 \quad (3.7)$$

For all $l \in L$, $t \in T$

If $t = 1$:

$$E_{l}^{l} = 1 - \sum_{i \in I_{spin}} \sum_{t' \in T, t < t' \leq t + \Delta T, l(i) = l} N_{i,t,t'} - \sum_{i \in I_{co}} \sum_{t' \in T, t < t' \leq t + \Delta T, l(i) = l} N_{i,t,t'}$$

If $1 < t < T_{max}$:

$$E_{l}^{l} = E_{l}^{l-1} + \sum_{i \in I_{spin}} \left( \sum_{t' \in T, t-\Delta T \leq t' < t, l(i) = l} N_{i,t,t'} - \sum_{t' \in T, t < t' \leq t + \Delta T, l(i) = l} N_{i,t,t'} \right)$$

$$E_{l}^{l} + \sum_{i \in I_{co}} \sum_{t' \in T, t < t' \leq t + \Delta T, l(i) = l} N_{i,t,t'}$$

If $t = T_{max}$:

$$E_{l}^{l} = E_{l}^{l-1} + \sum_{i \in I_{spin}} \left( \sum_{t' \in T, t-\Delta T \leq t' < t, l(i) = l} N_{i,t,t'} - \sum_{t' \in T, t < t' \leq t + \Delta T, l(i) = l} N_{i,t,t'} \right) + \sum_{i \in I_{co}} \sum_{t' \in T, t < t' \leq t + \Delta T, l(i) = l} N_{i,t,t'}$$

$$E_{l}^{l} \geq 0 \quad (3.8)$$
For all $s \in SP$, $t \in T$

If $t = 1$:

$$E_{sp}^s = spCap_s - \sum_{l \in \{PL_1, PL_5, PL_8, PL_9\}} 0.5 - \sum_{l \notin \{PL_1, PL_5, PL_8, PL_9\}} 1$$

If $1 < t < T_{max}$:

$$E_{sp}^s = E_{sp}^{s,t-1} + \sum_{l(i) \in \{PL_4, PL_5, PL_8, PL_9\}} 0.5 \cdot N_{i,t-1,t}^{co} + \sum_{l(i) \notin \{PL_4, PL_5, PL_8, PL_9\}} N_{i,t-1,t}^{co}$$

$$- \left( \sum_{l(i) \in \{PL_4, PL_5, PL_8, PL_9\}} 0.5 \cdot N_{i,t,t+1}^{co} + \sum_{l(i) \notin \{PL_4, PL_5, PL_8, PL_9\}} N_{i,t,t+1}^{co} \right)$$

If $t = T_{max}$:

$$E_{sp}^s = E_{sp}^{s,t-1} + \sum_{l(i) \in \{PL_4, PL_5, PL_8, PL_9\}} 0.5 \cdot N_{i,t-1,t}^{co} + \sum_{l(i) \notin \{PL_4, PL_5, PL_8, PL_9\}} N_{i,t-1,t}^{co}$$

$$- \left( \sum_{l(i) \in \{PL_4, PL_5, PL_8, PL_9\}} 0.5 \cdot N_{i,t,t+1}^{co} + \sum_{l(i) \notin \{PL_4, PL_5, PL_8, PL_9\}} N_{i,t,t+1}^{co} \right)$$

(3.11)

$$E_{sp,t} \geq 0$$

(3.12)

And finally, for all $t \in T$

If $t = 1$:

$$E_{ar}^t = arCap - \sum_{i \in I_{spin}} \sum_{t' \in T} \mu_{i,t'}^{ar} \cdot N_{i,t,t'}$$

If $1 < t < T_{max}$:

$$E_{ar}^t = E_{ar}^{t-1} + \sum_{i \in I_{spin}} \left( \sum_{t' \in T} \mu_{i,t'}^{ar} \cdot N_{i,t,t'} - \sum_{t' \in T} \mu_{i,t'}^{ar} \cdot N_{i,t,t'} \right)$$

(3.13)

If $t = T_{max}$:

$$E_{ar}^t = E_{ar}^{t-1} + \sum_{i \in I_{spin}} \sum_{t' \in T} \mu_{i,t'}^{ar} \cdot N_{i,t,t'}$$
If $t = 1$:

$$E_t^{dica} = dicaCap - \sum_{i \in I^{spin}} \sum_{t' \in T, t < t' \leq t + \Delta T} \mu_i^{dica} \cdot N_{i,t,t'}$$

If $1 < t < T_{max}$:

$$E_t^{dica} = E_{t-1}^{dica} + \sum_{i \in I^{spin}} \left( \sum_{t' \in T, t - \Delta T \leq t' < t} \mu_i^{dica} \cdot N_{i,t',t} - \sum_{t' \in T, t < t' \leq t + \Delta T} \mu_i^{dica} \cdot N_{i,t,t'} \right)$$

If $t = T_{max}$:

$$E_t^{dica} = E_{t-1}^{dica} + \sum_{i \in I^{spin}} \sum_{t' \in T, t - \Delta T \leq t' < t} \mu_i^{dica} \cdot N_{i,t,t'}$$

Where $E_{t}^{ar} \geq 0$ and $E_{t}^{dica} \geq 0$ (3.15)

In Equations 3.7, 3.13, and 3.14 only spinning tasks are considered, since only these consume the corresponding resources. However, for Equation 3.9 the changeover tasks are also considered, since the production lines are occupied during changeovers as well. On the other hand, in Equation 3.11 only the changeover tasks are considered, this because a die is occupied when it is installed in a production line, regardless of whether the specific line produces a product or not. More specifically, as soon as a die changeover begins, the die that is to be installed is occupied and it is freed by the end of the die changeover task that will replace this die with another one.

**Inventory level:** Similarly to the resource balance constraints, the inventory level $I_{p,t}$ of product $p \in P$ must always be at least zero. At the beginning of the scheduling horizon, there is an initial inventory level for each product $p$, denoted by $I_0^p$. For each event point $t \in T$, the inventory level of $p$ is calculated by considering the inventory level at the previous event point, and the amounts of the product that are produced or consumed at $t$. The corresponding equation is the following:

For all $p \in P$, $t \in T$

If $t = 1$:

$$I_{p,t} = I_0^p$$

If $t > 1$:

$$I_{p,t} = I_{p,t-1} + \sum_{i \in I^{spin}} \sum_{t' \in T, t - \Delta T \leq t' < t} \xi_{i,t',t} - \sum_{o \in O, p_{r(o)} = p} (dem_o \cdot p_{o,t} - p_{sl,o,t})$$

$$I_{p,t} \geq 0$$ (3.17)
Production Orders: As it was presented above, the due date \(d_{uo}\) of production order \(o \in O\) specifies the latest date that is possible to satisfy \(o\), either partially or fully. In other words, if \(o\) is (partially) satisfied at \(t\), the absolute time \(T_t\) cannot be later than \(d_{uo}\). This constraint is expressed by Equation 3.18. Another constraint regarding the production orders is that when an order is not satisfied at all, the value of \(p_{so,t}\) \(o\) must be zero. That way, the model is prevented from misusing the slack variable for increasing the inventory level, while no production order is satisfied. For this constraint, Equation 3.19 is used. Finally Equation 3.20 ensures that any production order can be satisfied at most at one event point.

For all \(o \in O\), \(t \in T\)
\[
T_t \leq d_{uo} + (1 - p_{so,t}) \cdot H
\]  
(3.18)
\[
0 \leq p_{so,t} \leq d_{mo} \cdot p_{so,t}
\]  
(3.19)

For all \(o \in O\)
\[
\sum_{t \in T} p_{so,t} \leq 1
\]  
(3.20)

Changeovers: For any production line \(l \in L\), when two recipes \(r, r' \in R\) that require a changeover time larger than zero, are performed in sequence, the suitable changeover task \(i' \in I^{co}\) must be scheduled between them. This is ensured by Equation 3.21. For any pair of event points \(t, t' \in T\), where \(t < t'\), and any pair of recipes \(r, r' \in R\) that require a changeover, a changeover task must be executed between \(t\) and \(t'\) if a task using \(r\) finishes at \(t\) and a task using \(r'\) starts at \(t'\), while no other spinning task is performed between \(t\) and \(t'\) at the same production line.

For all \(l \in L\), \(r, r' \in R\), \(t, t' \in T\), \(i' \in I^{co} | t < t', r^{co}(i') = (r, r')\) and \(l(i') = l\)
\[
\sum_{t'' \in T} N_{l'}^{i'}_{i''} - \sum_{t'' \in T} \sum_{t'' \in T} N_{i,t''} - \sum_{t'' \in T} \sum_{t'' \in T} N_{i,t''} = 1
\]  
(3.21)

Although Equation 3.21 considers most cases where a changeover task is needed, it does not take into account the cases where \(t = t'\). To deal with these cases Equation 3.22 is introduced, stating that for every production line \(l \in L\), there can be no pair of spinning tasks that require a changeover between them, with the second task starting at the event point when the first task finishes.

For all \(l \in L\), \(r, r' \in R\), \(t \in T\), \(i' \in I^{co} | t < T_{max}, r^{co}(i') = (r, r')\) and \(l(i') = l\)
\[
\sum_{t'' \in T} \sum_{t'' \in T} N_{i,t''} = 1
\]  
(3.22)

Besides Equations 3.21 and 3.22, some more constraints are needed to consider the recipes that were running on the production lines just before the beginning of the planning horizon. For each production line, before executing the first recipe, it must be checked whether a
changeover task needs to be scheduled or not. In Equation 3.23, for every production line $l \in L$, any pair of event points $t, t' \in T$, where $t < t'$, and any recipe $r \in R$ that can be performed on $l$ while a changeover task $i' \in I^{co}$ is needed between the initial recipe of $l$ and $r$, then this changeover task must be executed between $t$ and $t'$, if a spinning task $i \in I^{spin}$ on $l$ using $r$ starts execution on $t'$, while no other spinning task is scheduled on $l$ from the beginning of the planning horizon, until the beginning of task $i$.

\[
\sum_{t'' \leq t' < t'} \sum_{l' \in L, r' \in R} N_{t',l',t''} \geq \sum_{t'' \leq t' < t'} \sum_{i \in I^{spin}, r(i) = r} N_{l, t''} - \sum_{t'' \leq t' < t'} \sum_{i \in I^{spin}, l(i) = l} N_{l, t''} - \sum_{t'' \leq t' < t'} \sum_{i \in I^{spin}, l(i) = l} N_{l, t''} (3.23)
\]

Another constraint concerning the changeovers, is the non-technical Constraint 2, modeled by Equation 3.24. For every $w \in T_{we}$, every pair of subsequent event points $t, t + 1$, $t \in T$, where either of them occurs during weekend $w$ and for every production line $l \in L$, no changeover task $i \in I_{lo}$ of type die (i.e., $cht = 24$) with $l(i) = l$ can occur if it starts on $t$ and ends on $t + 1$. Similarly to Equations 3.3 and 3.5, Equation 3.24 is not linear, deteriorating the models performance.

\[
\sum_{i \in I^{co}, cht = 24} N_{l, t, t + 1} \leq 0 (3.24)
\]

Simultaneous production of specific product types: There are some product types that cannot be produced simultaneously with some other types. This constraint is modeled as shown below:

\[
\sum_{t', t'' \in T, t' < t'' \leq t' + \Delta T, r(i) = r} N_{l, t', r'} + \sum_{t', t'' \in T, t' < t'' \leq t' + \Delta T, r(i) = r'} N_{l, t', r} \leq 1 (3.25)
\]

### 3.4 Objective Function

The objective function is used to measure the quality of the solution given by the model. In this model, the objective function $OF$ comprises three terms and the optimal solution is derived through their minimization.

The first term of $OF$ is the non-delivery cost $O_{ndc}$. For a given product demand that has not been satisfied fully, $O_{ndc}$ represents the money that would have been earned if the full order was delivered. By minimizing this term, the difference between production need and actual production is minimized. For production order $o \in O$, the non-delivered amount is equal to the required product amount $(dem_o)$ minus the amount that has been delivered (if any). This amount is multiplied by the non-delivery costs $ndc_o$ of order $o$, given by Equation 2.1, to calculate the total non-delivery cost for the specific production
order.

\[ O_{ndc} = \sum_{o \in O} \left( \text{dem}_o - \sum_{t \in T} \left( \text{po}_{o,t} \cdot \text{dem}_o - \text{psl}_{o,t} \right) \right) \cdot \text{ndc}(o) \]  

(3.26)

The inventory deficit costs \( O_{idc} \) is the second term of \( OF \). Since the demand forecast is just an estimate of the actual product demand, AC tries to maintain stock of all products at specific levels. Hence, the model has to provide a solution where these inventory level targets (\( \text{stockTarget}_p \)) are satisfied as much as possible. To model this, when the stock of a product is below the corresponding target, a penalty is applied to the solution. In AC, the stock levels are only measured by the end of each month. For a particular month, the inventory deficit cost of product \( p \in P \) is equal to the stock deficit cost \( \text{stc}_p \) (representing the cost incurred per ton below the stock target), multiplied by the amount of tons below the stock target (measured exactly at the end of the month). Then, the total inventory deficit cost \( O_{idc} \) is equal to the summation of the monthly costs, for all products and all months of the planning horizon. The equation of inventory deficit cost, as given in Equation 3.27, is not linear. Unfortunately we did not manage to come up with a linear formulation. Hence, for performance purposes, instead of using the non linear equation in \( OF \), we decided to use Equation 3.28 which is linear, but calculates the inventory deficit cost at every time point of the scheduling horizon \( H \). Then, after finishing optimization, the non linear equation can be used to calculate the correct inventory deficit cost. The non linear formulation \( O_{idc} \) is given by the following equation:

\[ O_{idc} = \sum_{m \in \text{T}_{\text{month}}} \sum_{t \in T \cap m} \sum_{p \in P} \text{stc}_p \cdot \max((\text{stockTarget}_p - I_{p,t}), 0) \]  

(3.27)

And the linear formulation \( O_{idc}^* \) is:

\[ O_{idc}^* = \sum_{t \in T} \sum_{p \in P} \text{stc}_p \cdot \max((\text{stockTarget}_p - I_{p,t}), 0) \]  

(3.28)

Finally the last term of the objective function is the setup costs \( O_{sc} \). This corresponds to the total setup times performed during the whole planning horizon.

\[ O_{sc} = \sum_{i \in I_{\text{co}}} \sum_{t \in T} \sum_{T_{t} < t < T_{\text{max}}} (ch_{i,t} \cdot N_{i,t,t+1}^\text{co}) \]  

(3.29)

The summation of the aforementioned terms, multiplied by the corresponding coefficients, gives the objective function \( OF \):

\[ OF = W_{ndc} \times O_{ndc} + W_{idc} \times O_{idc}^* + W_{sc} \times O_{sc} \]  

(3.30)

Besides the objective function as described so far, a more general objective \( GO \) was also specified by AC in order to compare the different solutions of the model (on the same data instance). In this objective, besides the non delivery and inventory deficit costs, the cost of all changeovers is also considered by expressing it in euros. AC estimates that each changeover costs \( coCost \) euros irrespective of its type. So the equation is the following:

\[ O_{sc}' = \sum_{i \in I_{\text{co}}} \sum_{t \in T} \sum_{T_{t} < t < T_{\text{max}}} N_{i,t,t+1}^\text{co} \cdot coCost \]  

(3.31)

So, the general objective is equal to the following equation:

\[ GO = O_{ndc} + O_{idc} + O_{sc}' \]  

(3.32)
Note: For every modification or alternative that was tried in the model and is presented in the rest of the report, this single objective $GO$ is used to decide which alternative gives better results. In the rest of the report, the value of $GO$ is also called total costs.

### 3.5 Alternative Modeling of Constraints

After finishing with the development of the model and also during the evaluation phase (see Section 3.6 below), several constraints or variables were modeled in an alternative way, to investigate whether these would improve the model performance. This section presents these alternatives, while their effect on performance is shown in the following section along with the performance of the initial model.

**One time point at the end of each month:** The first constraint that we attempted to model differently is Equation 3.5, by getting a linear form for constraint $(T_t = m) = 1$, where $t \in T$, $m \in T_{month}$. To do so, the integer variables $a_{t,m}$ and the boolean variables $c_{t,m}$ are introduced. Variable $a_{t,m}$ must be equal to or greater than the absolute difference between $T_t$ and $m$, while $c_{t,m}$ is 1 when time point $t$ corresponds exactly to the end of month $m$ and 0 when $t$ corresponds to a date different than $m$. Then the following constraints are introduced:

For all $t \in T$, $m \in T_{month}$

\[
\begin{align*}
  a_{t,m} & \geq 0 \\
  a_{t,m} & \geq T_t - m \\
  a_{t,m} & \geq m - T_t \\
  c_{t,m} & \leq a_{t,m} \\
  H \times c_{t,m} & \geq a_{t,m}
\end{align*}
\]  

(3.33)

For all $m \in T_{month}$

\[
\sum_{t \in T} (1 - c_{t,m}) = 1
\]  

(3.34)

So, according to Equation 3.33:

- If $T_t = m$, then $a_{t,m} = 0$ and then $c_{t,m} = 0$
- If $T_t \geq m + 1$, then $a_{t,m} \geq 1$ and then $c_{t,m} = 1$
- If $T_t \leq m - 1$, then $a_{t,m} \geq 1$ and then $c_{t,m} = 1$

Hence it becomes clear that Equation 3.34 can only be true if and only if there is a single $c_{t,m}$ variable whose value is equal to 1, while the rest are equal to 0. In other words, there is a single time point assigned at the end of each month of the planning horizon.

**Die changeovers during weekends:** The next constraint that was modeled differently is the non-technical Constraint 2 that is initially expressed using Equation 3.24. The idea for the alternative formulation is quite similar to the previous one. The integer variables $a_{sp1}^{i,t,t+1}$, $a_{sp2}^{i,t,t+1}$, and the boolean variables $c_{sp1}^{i,t,t+1}$, $c_{sp2}^{i,t,t+1}$ are introduced, where $i \in I^{co}$ and $t \in T$, $t < T_{max}$. Variables $a_{sp1}^{i,t,t+1}$ must be 0 when the corresponding changeover overlaps
with the weekend and larger than 0 when the changeover ends before the weekend, while \( a_{i,t,t+1}^{sp1} \) must be 0 when the changeover overlaps with the weekend and larger than 0 when the changeover starts after the weekend. As to variables \( c_{i,t,t+1}^{sp1} \) and \( c_{i,t,t+1}^{sp2} \), they are 1, when the corresponding changeover overlaps with the weekend and 0 when there is no overlap. The constraints are defined as follows:

For all \( t \in T, \ i \in I^{co}, \ w \in T_{we} \mid t < T_{max} \land cht_i = 24 \)

\[
\begin{align*}
& a_{i,t,t+1}^{sp1} \geq 0 \\
& a_{i,t,t+1}^{sp1} > w - T_t - 24 \\
& e_{i,t,t+1}^{sp1} \neq c_{i,t,t+1}^{sp1} \\
& H \times (1 - c_{i,t,t+1}^{sp1}) \geq a_{i,t,t+1}^{sp1} \\
& a_{i,t,t+1}^{sp2} \geq 0 \\
& a_{i,t,t+1}^{sp2} > T_t - w - 48 \\
& e_{i,t,t+1}^{sp2} \neq c_{i,t,t+1}^{sp2} \\
& H \times (1 - c_{i,t,t+1}^{sp2}) \geq a_{i,t,t+1}^{sp2} \\
N_{i,t,t+1}^{co} \leq (1 - c_{i,t,t+1}^{sp1}) + (1 - c_{i,t,t+1}^{sp2})
\end{align*}
\] (3.35)

So according to the above, for a weekend starting at offset \( w \):

- If \( T_t < w - 24 \) (i.e. the changeover starts more than 24 hours before the weekend), then \( a_{i,t,t+1}^{sp1} \geq 1 \) and then \( c_{i,t,t+1}^{sp1} = 0 \). On the other hand, \( a_{i,t,t+1}^{sp1} = 0 \) and so \( c_{i,t,t+1}^{sp1} = 1 \).

- If \( T_t > w + 48 \) (i.e. the changeover starts after the end of the weekend), then \( a_{i,t,t+1}^{sp2} \geq 1 \) and then \( c_{i,t,t+1}^{sp2} = 0 \). On the other hand, \( a_{i,t,t+1}^{sp2} = 0 \) and so \( c_{i,t,t+1}^{sp2} = 1 \).

- If \( w - 24 \leq T_t \leq w + 48 \) (i.e. the changeover overlaps with the weekend), then \( a_{i,t,t+1}^{sp1} = 0 \) and \( a_{i,t,t+1}^{sp2} = 0 \) and then \( c_{i,t,t+1}^{sp1} = 1 \) and \( c_{i,t,t+1}^{sp2} = 1 \)

Thus when there is an overlap with a weekend, Equation 3.36 gives \( N_{i,t,t+1}^{co} \leq 0 \). However, if there is no overlap the equation becomes \( N_{i,t,t+1}^{co} \leq 1 \), allowing a die changeover.

**Starting production during weekdays:** The non-technical Constraint 1 is also modified in a similar way as the previous constraints. Similarly to before, the integer variables \( a_{i,t_1,t_2}^{sp3}, a_{i,t_1,t_2}^{sp4} \), and the boolean variables \( c_{i,t_1,t_2}^{sp3}, c_{i,t_1,t_2}^{sp4} \), where \( i \in I^{spin}, \ t_1, t_2 \in T, \) and \( t_1 < t_2 \leq t_1 + \Delta T \) are introduced. Variables \( a_{i,t_1,t_2}^{sp3} \) and \( a_{i,t_1,t_2}^{sp4} \) are larger than 0 when the task starts before and after the weekend respectively, and they are 0 when the task overlaps with the weekend. As to variables \( c_{i,t_1,t_2}^{sp3} \) and \( c_{i,t_1,t_2}^{sp4} \), they become 1 only when the task overlaps with the weekend.

For all \( t_1, t_2 \in T, \ i \in I^{spin}, \ w \in T_{we} \mid (t_1 < t_2 \leq t_1 + \Delta T) \land pr(i) \in weekPr \)
H \times (1 - c_{i,t_1,t_2}^{sp3}) \geq a_{i,t_1,t_2}^{sp3}

\begin{align*}
a_{i,t_1,t_2}^{sp4} & \geq 0 \\
a_{i,t_1,t_2}^{sp4} & > T_1 - w - 48 \\
c_{i,t_1,t_2}^{sp4} & \neq c_{i,t_1,t_2}^{sp4} \\
H \times (1 - c_{i,t_1,t_2}^{sp4}) & \geq a_{i,t_1,t_2}^{sp4}
\end{align*}

N_{i,t_1,t_2} \leq (1 - c_{i,t_1,t_2}^{sp3}) + (1 - c_{i,t_1,t_2}^{sp4})

(3.38)

So according to the above, for a weekend starting at offset w:

- If \( T_1 < w \) (i.e. the task starts before the beginning of the weekend), then \( a_{i,t_1,t_2}^{sp3} \geq 1 \) and then \( c_{i,t_1,t_2}^{sp3} = 0 \). On the other hand, \( a_{i,t,t+1}^{sp4} = 0 \) and so \( c_{i,t,t+1}^{sp4} = 1 \).

- If \( T_1 \geq w + 48 \) (i.e. the task starts after the end of the weekend), then \( a_{i,t_1,t_2}^{sp4} \geq 1 \) and then \( c_{i,t_1,t_2}^{sp4} = 0 \). On the other hand, \( a_{i,t_1,t_2}^{sp3} = 0 \) and so \( c_{i,t_1,t_2}^{sp3} = 1 \).

- If \( w \leq T_1 < w + 48 \) (i.e. the beginning of the task is about to overlap with the weekend), then \( a_{i,t_1,t_2}^{sp3} = 0 \) and \( a_{i,t_1,t_2}^{sp4} = 0 \) and then \( c_{i,t_1,t_2}^{sp3} = 1 \) and \( c_{i,t_1,t_2}^{sp4} = 1 \).

Thus when the start time of the task is about to overlap with a weekend, Equation 3.38 gives \( N_{i,t_1,t_2} \leq 0 \). However, if there is no overlap the equation becomes \( N_{i,t_1,t_2} \leq 1 \), allowing task \( i \) to be scheduled.

Note that the above hold, only if variables \( a_{t,m}^{sp}_{i,t_1,t_2}, a_{i,t,t+1}^{sp1}, a_{i,t,t+1}^{sp2}, a_{i,t_1,t_2}^{sp3}, \) and \( a_{i,t_1,t_2}^{sp4} \) are always assigned the smallest possible value. For example, if \( a_{t,m} = 0 \) whenever \( a_{t,m} \geq 0 \). According to the tests that we performed, this seems to be the case for the MIP model, although CPLEX gives in general extreme values (i.e. either the smallest or the biggest possible) in such cases. Since the MIP model had a very bad performance and due to time limitations, it was not investigated any further. However, if these equations need to be used in the future, more thorough testing has to be performed.

**Use floats for \( T_1 \):** A final alternative that was tried for improving the MIP model, was to use floats instead of integers for the decision variables \( T_1, t \in T \). The reasoning behind this idea is that it would be easier for CPLEX to solve the problem using floats instead of integers for \( T_1 \) and that it would not be too difficult to round the \( T_1 \) variables after obtaining the solution, just by making all scheduled tasks a little bigger or smaller.

### 3.6 Model Evaluation

Throughout the development of the MIP model, various data instances were tested to verify the model’s correctness and evaluate its performance. Initially some very simple data instances were used, to check all constraints one by one and make sure that the produced solutions are correct. Indeed the model proved to work as intended and no constraints were violated. As a next step, a little bigger data instances were tested to investigate how the model behaves in different scenarios (e.g. in cases of over- and under-capacity). Unfortunately, although the model gives correct solutions, its performance is
very low and it has problems for even small data instances. The following subsections present the main testing data instances that were used and give a description of the results obtained by the original model and the model with the alternatives of Section 3.5. Each alternative was tested separately to investigate whether it improves performance. All tables below give the running time that was required to prove optimality in each case. Furthermore, the number of model variables are provided as a measure of the data instance complexity. For each case, the number of variables is given in the form $(\alpha, \beta, \gamma)$, where $\alpha$ represents the number of integers, $\beta$ the number of floats, and $\gamma$ the number of binaries. Also, Alt. 1 refers to the alternative formulation regarding the assignment of one time point at the end each month, while Alt. 2 and Alt. 3 refer to the alternative formulations for the die changeovers during the weekends and the use of floats for decision variables $T_t, t \in T$, respectively. As to the alternative for the constraint about the starting of production during the weekdays, it is referred as Alt. 4. Please note that the very small and simple examples that were used initially to check the correctness of the constraints, are not presented here. Also note that in the following examples, setup costs are removed from the objective function in an attempt to make tests simpler.

3.6.1 1 Product, 1 Production Line

This is one of the main examples that were tested on the MIP model. The data instance consists of a single production line and a single product along with all the different recipes of producing it. Three variants of this data instance were tested, where the first consisted of a single demand of just 100 tons by the end of the planning horizon $H$. The purpose of this test was to investigate the model behavior in cases of over-capacity. The model gave the optimal solution (in terms of the objective function), by choosing to use the recipe that was already on the production line and not the one with the highest throughput. Thus it avoided a die changeover, but the production finished far later than it would finish if the model would choose to perform a changeover and execute the fastest recipe. However the demand was fully satisfied and since neither the completion time nor the changeover costs were objectives, both solutions are optimal.

The second variant contained an “infinite” demand (1000 tons) whose deadline was again $H$. In this scenario of under-capacity it was tested whether the model is able to distinguish which recipe is best for delivering the maximum demand amount possible. Indeed, the model gave the optimal solution by choosing the recipe with the highest production throughput and it used this (after the required changeover) during the whole planning horizon to produce as much product as possible. Another possible solution would be to choose the recipe that was already on the production line and thus it would avoid the changeover. However, by doing so it would end up producing a smaller amount of product and thus the final solution would not be optimal.

Finally, the third variant consisted of 3 demands in order to test if the model can handle multiple demands of the same product. For each demand there was a different deadline and also the required amounts varied. Again the model gave the optimal solution by managing to handle all demands successfully.

Even though the solutions from all tests were correct and optimal, it was disappointing to find out that it took the original model a lot of time to prove optimality. Although the model spent only a few seconds (approximately 1.5) to find the solution and prove optimality for the first variant, for the last two variants of the data instance the required time was quite high. More specifically, in both cases the best solution was found quite fast, but then the model needed approximately 65 seconds for the second variant and 79 for the third to prove that the particular solution was the optimal one. The alternative formulations were then applied separately to investigate whether they improve performance. Table 3.1 presents the results obtained by the original model and all the alternatives.
As shown in the table, it is clear that Alt. 2 performs worse than the original model in all variants of the data instance. As to the other two alternatives, it seems that they don’t contribute to the model performance either, since they don’t perform better in all variants. Also note the increase in the number of variables. This was expected, since new variables are introduced in Section 3.5, in order to linearize the non linear equations.

Regarding Alt. 4, the second variant was used again with the difference that the first two days of the planning horizon belonged to a weekend and that the product under consideration was only allowed to start production during the weekdays. The running time until proving optimality was 6.47 seconds for the original model and 69.03 when using Alt. 4. As to the number of variables, for the original model they were (10, 474, 244) (same as in Table 3.1 for variant 2), while for the alternative model they were (298, 474, 532).

The results indicate that it may be preferable to let CPLEX translate the non-linear equations to linear, instead of introducing extra variables. What is made clear though, is that the performance of the MIP model is quite poor even for very small data instances, since it cannot prove optimality fast. Of course, the model can be stopped before proving optimality and the results can still be used. However, this is a first indication that the current formulation of the MIP model is not appropriate for large data instances, which becomes clearer in the following more complex examples.

### 3.6.2 2 Products, 1 Production Line

The purpose of the second data instance was to investigate how the model behaves when it needs to consider more than one product. Again only one production line was available while there were two products, namely A and B. For both products there was a single demand of 1000 tons each, with the difference that the contribution of A was slightly higher. That way it would be checked whether the model is able to identify which demand is most important (even slightly) and has a priority over the others. The model chose correctly to produce only product A using the recipe with the highest throughput, while totally neglecting product B. The model also chose to do the necessary changeover for this.

In the second variant of the instance, both products had the same non delivery cost per product unit and an equal production throughput for all recipes. In this case the model produced product B instead of A, since no changeover was needed. Again the model made the correct decision among the two different products. However, optimality was not proven in neither of the two cases even after running for 500 seconds. Table 3.2 shows the results of the corresponding tests. Here instead of the running time, the gap between the best solution found and the lower bound is provided as a percentage, when the model runs for 500 seconds. The lower bound is actually a value for which it is proven that there is no solution having a better objective value.

Again, as shown in the results of these instances, it is not clear which alternative formulation performs better, since neither of them is better than the original in both variants of the data instance. In all the cases presented in Table 3.2, optimality would
eventualy be proved if the model was let to run for more time. However, it was stopped at 500 seconds, since this amount of time is considered already too long for such a simple data instance.

### 3.6.3 2 Products, 3 Production Lines

A somewhat more complex data instance consisted of two different products and three production lines. \( \text{contribution}_p \) was the same for both products and they both had a single demand of 1000 tons due by the end of the planning horizon. The purpose of this instance was to test whether the model was able to handle multiple production lines with several possible recipes per line. Unfortunately this instance proved to be quite complex for the model, as the original model found a first solution only after 460 seconds. Then the model did not manage to prove optimality even after running for 1000 seconds. As to the alternatives, they all found a first solution faster than the original model and after 1000 seconds, the gap was smaller than when running the original model. However, optimality was still not proven within 1000 seconds. Table 3.3 presents the results for the original model and all alternatives.

<table>
<thead>
<tr>
<th></th>
<th>Variant 1</th>
<th>Variant 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gap</td>
<td>Number of variables</td>
</tr>
<tr>
<td>Original</td>
<td>64.4%</td>
<td>(10, 628, 1406)</td>
</tr>
<tr>
<td>Alt. 1</td>
<td>60.28%</td>
<td>(40, 628, 1436)</td>
</tr>
<tr>
<td>Alt. 2</td>
<td>65.17%</td>
<td>(2206, 628, 3602)</td>
</tr>
<tr>
<td>Alt. 3</td>
<td>65.03%</td>
<td>(0, 638, 1406)</td>
</tr>
</tbody>
</table>

Table 3.3: Model performance for 2 products and 3 production lines, after running for 1000 seconds

In this more complex data instance, it becomes clear that the alternative formulations of the non linear constraints contributes to the performance improvement. However, still the model does not perform well enough for such a small data instance. So it becomes evident that the current MIP model is not able to handle efficiently the problem of AC at its full complexity. More specifically, the data instance in these tests uses only 2 products, 3 production lines, and 2 product demands, while a full data instance contains approximately 50 products, 11 production lines and 150 demands.

### 3.6.4 April, May, June 2011 Data Instance

Finally the instance of April, May, June 2011 was tested although there were no expecta-tions of getting any result. In fact CPLEX did not manage to run neither the original model nor the model with any of the alternative formulations. Actually, the model did not
even manage to create the set of changeover tasks $I^{co}$. Hence the exact number of binary variables cannot be calculated, but they are certainly more than 20000. Besides binaries, the original model requires 18148 float and 10 integer variables. Of course, the 3-months problem could be decomposed in 3 separate single month problems in order to reduce complexity. However, CPLEX is not expected to handle this either (at least not efficiently), since the model failed to handle successfully even the data instance of Section 3.6.3.
Chapter 4

Constraint Programming Model

Besides the MIP model that is described in Chapter 3, we implemented a CP model as well. We decided to do so, because of the unsatisfactory performance of the MIP model. Modeling the same problem using two different approaches, showed that in general it was easier to model the constraints of AC using CP. Especially the verification of whether some particular constraints were modeled correctly (like the constraints for changeovers), was easier in CP. Performance comparisons between the two approaches are not possible, since the MIP model does not even manage to run the full data instances.

The input and output of the model are given in Sections 4.1 and 4.2. The various constraints are formulated as shown in Section 4.3, and then the objective function is described in Section 4.4. Afterwards, several modifications are presented in Section 4.5 that either improved the model performance or (in contrast to our expectations) they did not. Finally, Section 4.6 describes how the model imports fixed productions, maintenance stops, and/or production line shut downs to the solution, that are required by the real data instances, while Section 4.7 gives the final configuration of the model.

4.1 Input

Regarding the input of the CP model, it is exactly as described in Section 2.4. The only difference is that although the array of changeover durations is provided to the CP model, no set of possible changeover tasks is given, this because changeovers are modeled in a different way.

4.2 Output

Although most input elements are the same for both models, the output elements are different in the CP model. This because in CP the model result is given in a different form than in MIP. Let monthRange denote the set of months in the planning horizon $H$. In the CP model, the solution is given as a sequence seg of intervals per production line $l \in L$, where each interval $spt_{i,m}$ represents the execution of spinning task $i \in I^{\text{spin}}$ on $l$ during month $m \in \text{monthRange}$. Intervals are decision variables used in CP for scheduling, representing intervals of time during which the corresponding tasks are executed. An interval $g$ has a start time $st_g$, an end time $et_g$, and a duration $d_g$, where $st_g + d_g = et_g$. For each spinning task $i \in I^{\text{spin}}$ and month $m \in \text{monthRange}$, a single interval $spt_{i,m}$ is created, whose earliest start date is the beginning of month $m$ and its latest end date is the end of $m$. Thus any spinning task $i$ can be scheduled at most once per month. In addition, each interval has a minimum duration of $mpt_{i,(i)}$ and a maximum duration equal to the length of the month. Finally, these intervals are optional, meaning that it is not mandatory to be scheduled. Each interval $spt_{i,m}$ is associated to a type, denoted by $intType_{spt_{i,m}}$. 

31
This type is equal to the id of the recipe \( r \in R \) used by \( i \in I^{\text{spin}} \) of the corresponding interval and, as it is shown below, it is used in various constraints. Besides the intervals for spinning tasks, changeover tasks are also modeled as optional intervals and they are part of the solution output as well. To model these changeovers, for each \( spt_{i,m} \) interval, a \( cot_{i,m} \) interval is created whose length can be 0, 3, 7, or 24 hours and is present if and only if the corresponding \( spt_{i,m} \) is present as well. Note though that the changeover tasks \( i' \in I^{\text{co}} \) used in the MIP model are not used in this model. Of course, instead of having present changeover task intervals of 0 length, they could just not be present. However, this would result in a worse way of modeling the corresponding constraints, since \textit{meta-constraints} should be used. Meta-constraints are constraints that are built by combining constraints through logic. For constraints \( C_1, C_2 \), an example is \( C_1 \Rightarrow C_2 \), meaning that if \( C_1 \) holds, then \( C_2 \) must also hold. In general meta-constraints perform worse than normal ones, and thus they should be avoided. Another output element is the inventory level of each product \( p \in P \), which is given only by the end of each month. For a specific product \( p \) and a month \( m \), the inventory level \( I_{p,m} \) is equal to the inventory level of the previous month \( (m-1) \), plus the total product amount that was produced during the current month, minus any amount that is delivered to customers, satisfying orders \( o \in O \) of product \( p \). Finally, the output provides the amount of product that was delivered for each product order \( o \), denoted by the demand satisfaction variable \( ds_o \). For a given \( o \), if \( ds_o = 0 \), it means that the order was not satisfied at all, while \( ds_o = dem_o \) means that the total demand was satisfied. To sum up, the decision variables mentioned so far are the spinning task and changeover task intervals \( spt_{i,m} \) and \( cot_{i,m} \), where \( i \in I^{\text{spin}} \) and \( m \in \text{monthRange} \), the inventory level \( I_{p,m} \) of product \( p \in P \) on month \( m \), the demand satisfaction amount \( ds_o \) of order \( o \in O \) and the sequence \( seq_l \) of spinning tasks on production line \( l \in L \).

### 4.3 Constraints

The constraints of the CP model are the same with the MIP model. However, the way that they are modeled is different. Since the CP model was developed using the Optimization Programming Language (OPL) and the CP Optimizer (CPO) engine of IBM ILOG CPLEX Optimization Studio, all constraints are modeled according to the syntax used in OPL.

**No overlapping productions and changeover times:** The most simple and fundamental constraint is that each production line \( l \in L \) can execute only one spinning task \( i \in I^{\text{spin}} \) at any given time. This constraint is modeled using the \textit{noOverlap} constraint on the sequence \( seq_l \) of spinning task intervals of line \( l \). The second (optional) argument provided to the \textit{noOverlap} constraint is a transition matrix \( D \). The \textit{noOverlap} constraint on the interval sequence variable \( seq_l \) states that \( seq_l \) defines a chain of non-overlapping intervals. Furthermore, when the transition matrix \( D \) is passed to the constraint, it means that if interval \( spt_{i',m'} \) appears just after interval \( spt_{i,m} \), where \( i, i' \in I^{\text{spin}}, m, m' \in \text{monthRange} \), then a minimum distance \( D(\text{intType} spt_{i,m}, \text{intType} spt_{i',m'}) \) must be respected between the end of \( spt_{i,m} \) and the beginning of \( spt_{i',m'} \). That way the minimum changeover times are also considered by this constraint. The \textit{noOverlap} constraint has the following definition:

\[
\text{For all } l \in L
\]

\[
\text{noOverlap}(seq_l, D) \quad (4.1)
\]

**Changeover intervals:** Although the required changeover times are modeled by the \textit{noOverlap} constraint, the changeover tasks are modeled as intervals as well since they
are needed for some other constraints presented below. Thus several constraints are used to schedule the right changeover intervals with the right duration at the right time point. First of all, as said before, for any \( i \in I^{\text{spin}} \) and any \( m \in \text{monthRange} \) the \( \text{cot}_{i,m} \) interval is present only if \( \text{spt}_{i,m} \) is present as well. Furthermore, \( \text{cot}_{i,m} \) must be scheduled to finish exactly when \( \text{spt}_{i,m} \) starts. These constraints are modeled as shown in Equations 4.2 and 4.3 using the \text{presenceOf} and \text{endAtStart} constraints respectively. For any pair of intervals \( f \) and \( g \), the constraint \text{presenceOf}(f) \) states that interval \( f \) must be present, while constraint \text{endAtStart}(f, g) \) states that if both \( f \) and \( g \) are present, then \( g \) must start exactly when \( f \) ends.

For all \( i \in I^{\text{spin}}, \ m \in \text{monthRange} \)
\[
\text{presenceOf}(\text{cot}_{i,m}) = \text{presenceOf}(\text{spt}_{i,m})
\] (4.2)

\[
\text{endAtStart}(\text{cot}_{i,m}, \text{spt}_{i,m})
\] (4.3)

Besides Equations 4.2 and 4.3 there is an additional constraint used for scheduling the changeover tasks. Equation 4.4 defines the length of \( \text{cot}_{i,m} \), where \( i \in I^{\text{spin}} \) and \( m \in \text{monthRange} \), using the \text{lengthOf} constraint, which determines the length of an interval. The length is equal to the required changeover duration \( ct(r, r') \) between recipes \( r, r' \in R \) as defined in Section 2.4. For interval \( \text{spt}_{i,m} \) of sequence \( \text{seq}_i \), the expression \( \text{typeOfPrev}(\text{seq}_i, \text{spt}_{i,m}, \text{curRec}_{i(i)}) \) returns the type of interval \( \text{spt}_{i,m}, i' \in I^{\text{spin}}, \) scheduled in \( \text{seq}_i \) just before \( \text{spt}_{i,m} \). In case interval \( \text{spt}_{i,m} \) is the first interval of the sequence, then \( \text{typeOfPrev}(\text{seq}_i, \text{spt}_{i,m}, \text{curRec}_{i(i)}) \) returns \( \text{curRec}_{i(i)} \) which is the recipe of the corresponding production line just before the beginning of the scheduling horizon.

For all \( i \in I^{\text{spin}}, \ m \in \text{monthRange} \)
\[
\text{lengthOf}(\text{cot}_{i,m}) = ct(\text{typeOfPrev}(\text{seq}_i, \text{spt}_{i,m}, \text{curRec}_{i(i)}), r_i)
\] (4.4)

A constraint regarding the changeover intervals is Constraint 2 of the non-technical constraints, stating that no die changeover is allowed to occur during a weekend. To model this, for each changeover interval four new intervals are created representing the type of the changeover (soft, normal, block, and spin changeover). So for any \( \text{cot}_{i,m} \) interval, where \( i \in I^{\text{spin}} \) and \( m \in \text{monthRange} \), the intervals \( \text{cotType}_{i,m,t} \) are created, where \( t \in \{0, 3, 7, 24\} \) denotes the type of the changeover according to its length. Similarly to the other intervals, \( \text{cotType}_{i,m,t} \) is optional and its size is equal to \( t \). When a changeover interval \( \text{cot}_{i,m} \) is present, only the \( \text{cotType}_{i,m,t} \) whose type \( t \) is the same as the type of the changeover must be present too. This is ensured by the \text{alternative} constraint, which states that for all \( i \in I^{\text{spin}} \) and \( m \in \text{monthRange} \), if \( \text{cot}_{i,m} \) is present then exactly one of the \( \text{cotType}_{i,m,t} \), \( t \in \{0, 3, 7, 24\} \) is present starting and ending exactly when \( \text{cot}_{i,m} \) starts and ends as well. Equation 4.5 shows how \text{alternative} is used. Note the \text{all} expression, used to include all four types of changeovers.

For all \( i \in I^{\text{spin}}, \ m \in \text{monthRange} \)
\[
\text{alternative} (\text{cot}_{i,m}, \text{all}(t \in \{0, 3, 7, 24\}) \text{cotType}_{i,m,t})
\] (4.5)

After specifying the \( \text{cotType}_{i,m,t} \) intervals, non technical Constraint 2 is modeled by means of the \text{forbidExtent} constraint, indicating that a given interval cannot overlap with a specific date. These specific dates are specified using \text{stepwise linear functions} (i.e. piecewise linear functions where all slopes are 0). If \( \text{TWE} \) denotes a stepwise function which is 100 (maximum value) during the weekdays and 0 (minimum value) during the weekends, the constraint is formulated as follows:

For all \( i \in I^{\text{spin}}, \ m \in \text{monthRange}, t \in \{0, 3, 7, 24\} \)
\[
\text{forbidExtent}(\text{cotType}_{i,m,t}, \text{TWE})
\] (4.6)
**Resource balance constraints:** As already mentioned, each resource has a specific capacity which cannot be exceeded. This set of constraints is modeled using cumulative functions, which model the cumulated usage of a resource as a function of time. For a given resource $e$, the required amount at any time point is given by the cumulative function $E_e$. By the time a spinning task interval $spt_{i,m}$, where $i \in I_{spin}$, $m \in monthRange$, starts execution, $E_e$ is increased by the amount $\mu_e$ of resource $e$ required by spinning task $i$. Similarly, when the execution of $i$ is terminated $E_e$ is decreased by the same amount $\mu_e$. Hence, if $E_{hr}^h$, $E_{ar}^h$, $E_{dica}^h$, and $E_{sp}^s$ denote the cumulative functions for the high-rise, acid evaporation, dica, and die type resources respectively, where $s \in SP$ and $h \in HR$, the following equations are defined:

For all $h \in HR$
\[ E_{hr}^h \leq hrCap_h \]  \hspace{1cm} (4.7)

\[ E_{ar}^h \leq arCap \]  \hspace{1cm} (4.8)

\[ E_{dica}^h \leq dicaCap \]  \hspace{1cm} (4.9)

For all $s \in SP$
\[ E_{sp}^s \leq spCap_s \]  \hspace{1cm} (4.10)

**Inventory level and demand satisfaction:** In Section 4.2 it is explained that for $p \in P$, $m \in monthRange$, and $o \in O$ variable $I_{p,m}$ denotes the inventory level of product $p$ by the end of month $m$, while $ds_o$ denotes the satisfied amount of production order $o$. Both are defined as integer non-negative variables and $ds_o$ has an upper limit equal to the production order amount $dem_o$. For each product $p$, the inventory level at the end of the first month $m$ is equal to the initial inventory level $I_{p,m}^0$ of $p$, plus the amount of $p$ that was produced during the current month, minus any amount that was consumed for satisfying demand $o$. However the inventory level $I_{p,m}$ of $p$ at the end of a month $m > 1$ is equal to the inventory level of the previous month, plus the amount of $p$ that was produced during the current month, minus any amount that was consumed for satisfying demand $o$. The amount produced by an interval $spt_{i,m}$, where $i \in I_{spin}$ and $m \in monthRange$, is equal to its length multiplied by the production rate $thp_{r(i)}$. So the following constraint is used:

For all $p \in P$, $m \in monthRange$
If $m = 1$:
\[ I_{p,m} = I_{p,m-1} + \sum_{i \in I_{spin}} lengthOf(spt_{i,m}) \cdot thp_{r(i)} - \sum_{o \in O} ds_o \]  \hspace{1cm} (4.11)

If $m > 1$:
\[ I_{p,m} = I_{p,m-1} + \sum_{i \in I_{spin}} lengthOf(spt_{i,m}) \cdot thp_{r(i)} - \sum_{o \in O} ds_o \]
Initial changeovers: In order to consider the changeovers that are required at the beginning of the planning horizon $H$, the $startOf$ constraint is used that determines the starting time of an interval. The starting time of every interval $spt_{i,m}$, where $i \in I^{spin}$ and $m \in monthRange$, must be greater than or equal to the changeover time needed between the initial recipe of line $l \in L (curRec_l)$ and the recipe $r(i)$ executed by $i$. In the $startOf$ constraint, an integer is provided as a second (optional) argument. This integer is used as a return value of $StartOf$, when interval $spt_{i,m}$ is absent. Thus, as a second argument in the $startOf$ constraint is given the length of the planning horizon $H$, which is definitely larger than any possible changeover. Then, the constraint for the initial changeovers is as follows:

For all $i \in I^{spin}, m \in monthRange$
\[
\text{startOf}(spt_{i,m}, H) \geq ct(curRec_{l(i)}, r(i))
\] (4.12)

Simultaneous production of specific product types: To ensure that the simultaneous production of some products is not allowed, the $startOf$ constraint is used, as well as the $endOf$ constraint. Similarly to $startOf$, $endOf$ is used to specify the ending time of an interval $g$. The definition of the constraint is given in Equation 4.13 below and it states that if $spt_{i,m}$ and $spt_{i',m}$, where $i, i' \in I^{spin}$ and $m \in monthRange$, are two intervals that are not allowed to overlap, then $spt_{i,m}$ must either end before $spt_{i',m}$ starts or start after $spt_{i',m}$ ends.

For all $i, i' \in I^{spin}, m \in monthRange | l(i) \neq l(i'), \neg \allowSim_{r,r'}$
\[
\text{endOf}(spt_{i,m}) \leq \text{startOf}(spt_{i',m}) \lor \text{startOf}(spt_{i,m}) \geq \text{endOf}(spt_{i',m})
\] (4.13)

Production initialization: Non technical Constraint 1 states that the production of some specific products can only start during the weekdays. For this constraint the $TWE$ stepwise function is used again. However instead of the $forbidExtent$, the $forbidStart$ constraint is used. For interval $spt_{i,m}$, where $i \in I^{spin}$ and $m \in monthRange$, the constraint $forbidStart(spt_{i,m}, TWE)$ states that $spt_{i,m}$ cannot start when $TWE$ is equal to zero.

For all $i \in I^{spin}, m \in monthRange | pr_i \in \{P_1, P_2, P_3\}$
\[
\text{forbidStart}(spt_{i,m}, TWE)
\] (4.14)

4.4 Objective Function

The objective function used in the CP model is the same as the one presented in Section 3.4. The equations are adjusted accordingly and they are presented below.

$O_{ndc} = \sum_{o \in O} (dem_o - ds_o) \cdot ndc_o$ (4.15)

$O_{idec} = \sum_{p \in P} \sum_{m \in monthRange} stc_p \cdot \max ((stockTarget_p - I_{p,m}), 0)$ (4.16)

$O_{sc} = \sum_{i \in I^{spin}} \sum_{m \in monthRange} lengthOf(cot_{i,m})$ (4.17)

$O'_{sc} = \sum_{i \in I^{spin}} \sum_{m \in monthRange} (lengthOf(cot_{i,m}) > 0) \cdot coCost$ (4.18)
4.5 Alternative Modeling and Improvement Modifications

The performance of the CP model as described so far had a low performance. Initial tests showed that the produced solution consisted of small intervals only, with many gaps on all lines. Obviously the values of the objectives were very high as well. To tackle this, several attempts have been made towards improving the model. The inspiration of the various modifications or alternatives that were attempted came either by examining the solution or by searching the code for any badly modeled elements or constraints. We furthermore used the literature study presented in Section 1.3, since the MIP/CP decomposition described in Section 4.5.1, was proposed by several papers. All modifications that contributed to the improvement of the model are presented in this section, along with some others that brought worse results although they were expected to improve the behavior of the model.

The data instances used for testing are the ones presented below. In all cases, the tests that are considered for deciding whether a modification is beneficial or detrimental, are the ones performed on the two 3-months data instances. This because in AC’s practice 3-months periods are considered. However, the single month data instances are also tested, to get a better understanding of a modification’s effect. The data instances used are the following single month instances:

- April 2011
- June 2012
- August 2012

and the following 3-months instances:

- April, May, June 2011
- June, July, August 2012

Unfortunately, no more data instances were tested due to time limitations. Note that although the instances of April 2011 and April, May, June 2011 are old, they were chosen to test the model in cases of under-capacity. As to the two single month data instances of 2012, they were chosen because of the fixed tasks that they contained.

The following list contains all the modifications that were included while testing a proposed modification on any data instance. The modifications of the list, are the ones that we were expecting to improve the model’s performance, or that we already knew that they had a positive effect. Hence we included them, to investigate how each particular modification affects the performance of the (possibly) final model. So, all tests are performed using the following modifications:

- Size decomposition is applied (Section 4.5.1)
- The sequence search phases is used (Section 4.5.2)
- A maximum inventory level is set, equal to 120 tons per product (Section 4.5.3)
- Equation 4.25 is included for the data instance of April, May, June 2011, while Equation 4.23 is included for the data instance of June, July, August 2012 (Section 4.5.4)
- Equation 4.12 is replaced by Equation 4.26 (Section 4.5.5)
- Equation 4.27 is included in the model (Section 4.5.6)
- The minimum processing time $m_{pt}$ is raised from 120 to 156 hours (Section 4.5.7)
• The post processing method is used (Section 4.5.8)

• For the data instance of June, July, August 2012 where fixed tasks are present, Equation 4.31 is used (Section 4.6)

Note that all single month data instances are run for 100000 fails (i.e., after 100000 failures, the algorithm is terminated) and all 3-months instances for 3000 seconds. Finally, on all tests the weight factors of the objective function are the following:

- $W_{ndc} = 100$
- $W_{idc} = 1$
- $W_{sc} = 0.05$

4.5.1 Model decomposition

Size decomposition

Although the model (as described in the previous sections) produces correct results, a data instance of 3 months is too large and the model cannot handle it. More specifically, for the data instance of April, May, June 2011, the model cannot start running and COS gives several errors about the model complexity. For the data instance of June, July, August 2012, although the model runs, it takes too long to give an initial solution. Thus it was decided to decompose the problem into several smaller ones.

When a problem (or data instance) is too complex to solve, decomposition can have a great impact. In the case of AC, instead of having a model that optimizes the schedule of three months in one run, each month is optimized separately. Hence, by running the model 3 times iteratively and then combining the results, the schedule of a 3 months period is created, whose quality is far better than the quality of the original model solution.

Besides the performance improvement, this kind of decomposition has additional benefits. First of all the model is smaller. Hence, if more constraints need to be included in the future, the model will be able to handle them more easily.

Furthermore, in the current version of the model, during each iteration the sub-model saves the solution to a database. Then in the next iteration, the sub-model “reads” the data needed for the next month by that database and starts solving. That way the solution of a specific month can be used again and again without needing to rerun it. This proved to be very helpful and a great time saver especially during testing where each iteration was tested several times with different parameters.

Finally, another advantage of this decomposition method is that now it is possible to run the model for as big data instances as needed. Before introducing decomposition, if the data instance provided in the model comprised of many months, it would imply an increased number of variables for the additional intervals needed for both spinning and changeover tasks as well as the extra production orders and inventory level variables. Thus the model would become larger and even if it would still be able to run, its performance (per month) is expected to be degraded considerably. However, after introducing decomposition, each month is independent of the number of months that will be considered in total. Thus the performance of each month is not affected by the total length of the planning horizon.

Besides the advantages, there is an important disadvantage as well. While the model is running for a specific month, it does not consider the data (especially the forecasts) of the upcoming months. That way, the model may take a decision that although it is beneficial for the current month, it can be so detrimental for the solution of the following months that the overall effect will be negative. An example of this disadvantage is shown in Section 5.3.
Problem decomposition

Another kind of decomposition applied to the initial model, is to solve the problem by using a combination of two models. At first a MIP model runs, deciding only on the amount of products that will be produced on each line per month and also the recipes that will be used. Then the CP model takes over, using the MIP solution as a starting point, deciding on the exact schedule. As it is mentioned in Chapter 1, both MIP and CP based methods have strengths and weaknesses, and if they are combined they can help to solve problems that would otherwise be impossible or very hard to solve. Hence, this kind of decomposition was tried, since it was expected that the combination of MIP and CP would be beneficial for the problem of AC.

The CP model used in this decomposition method is the same as described so far (including size decomposition). However the MIP model is quite simple as it considers only a few constraints. The first constraint contains classified information and thus it is not presented in this version of the report.

The next constraint that is considered by the MIP model is the non-technical Constraint 3 of Section 2.3. If a task $i$ is present, its length must be at least equal to the minimum production time $mpt$. To model this constraint, the binary variables $c_{i,m}, i \in I^{spin}, m \in monthRange$ are introduced, which become 1 when the length of task $i$ is 0 (i.e., the task does not run). Then the following equation is used:

For all $i \in I^{spin}, m \in monthRange$

$$H \times (1 - c_{i,m}) \geq len_{i,m}$$

$$c_{i,m} \neq len_{i,m}$$

$$c_{i,m} \times mpt + len_{i,m} \geq mpt$$

(4.19)

From Equation 4.19 it follows that:

- If $len_{i,m} = 0$, then $c_{i,m} = 1$ and so the last equation gives $mpt \geq mpt$, which always holds.
- If $len_{i,m} > 0$, then $c_{i,m} = 0$ and so the last equation gives $len_{i,m} \geq mpt$.

That way it is ensured that any scheduled task has an execution time at least equal to the minimum allowed production time $mpt$.

Then a constraint for calculating the total changeover duration on line $l \in L$ is introduced. Since the production sequence of each line is not determined, a degree of pessimism is introduced again. For each task $i \in I^{spin}$, the model considers a changeover of maximum duration $maxCo$. Equation 4.20 presents the particular constraint:

For all $i \in L$

$$coLen_l = \sum_{i \in I^{spin} \atop \ell(i) = l} maxCo \times (1 - c_{i,m})$$

(4.20)

As to the inventory level and the demand satisfaction, a constraint similar to Equation 4.11 is used. The variables $I_{p,m}$, and $ds_o$, where $p \in P$, $m \in monthRange$, and $o \in O$,
are used in the MIP model too and the constraint is as follows:

For all $p \in P$, $m \in monthRange$

If $m = 1$:

$$I_{p,m} = I_{p,0}^0 + \sum_{i \in \{p \text{ spin}\}} len_{i,m} \times thp_r(i) - \sum_{o \in O \text{ pro}=p \text{ due}=m} ds_o$$  \hspace{1cm} (4.21)

If $m > 1$:

$$I_{p,m} = I_{p,m-1} + \sum_{i \in \{p \text{ spin}\}} len_{i,m} \times thp_r(i) - \sum_{o \in O \text{ pro}=p \text{ due}=m} ds_o$$

Finally, a last constraint is used for considering the die capacity. The total time duration that a particular die type is used on all lines, cannot be more than the capacity of that die type, multiplied by the length of the planning horizon $H$. Equation 4.22 presents how this constraint is modeled:

For all $s \in SP$

$$\sum_{i \in \{p \text{ spin}\}} len_{i,m} \leq H \times spCap_s$$  \hspace{1cm} (4.22)

As shown in Table 4.1, in general the MIP/CP model performs worse than the CP model in the small tests. An improvement of 2% is achieved for the data instance of April 2011, but still, this is achieved at a cost of considerably more running time (180 seconds in the CP model, 284 in the MIP/CP). As to the data instances of June 2012 and August 2012, the MIP/CP takes more time to reach the limit of 100000 fails and it also gives worse results. The fact that the CP model performs better is also shown in the tests on 3-months data instances, presented in Table 4.2. Hence, we conclude that the MIP/CP model as described in this section, does not improve performance.

<table>
<thead>
<tr>
<th>Model</th>
<th>Data instance</th>
<th>April 2011</th>
<th>June 2012</th>
<th>August 2012</th>
<th>Average percentage difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Run time (seconds)</td>
<td>Total cost (million)</td>
<td>Run time (seconds)</td>
<td>Total cost (million)</td>
</tr>
<tr>
<td>CP</td>
<td></td>
<td>180</td>
<td>46.03</td>
<td>103</td>
<td>7.75</td>
</tr>
<tr>
<td>MIP/CP</td>
<td></td>
<td>284</td>
<td>45.07 (-2.1%)</td>
<td>112</td>
<td>12.22 (+57.7%)</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison of CP and MIP/CP models (simple tests)

<table>
<thead>
<tr>
<th>Model</th>
<th>Data instance</th>
<th>April, May, June 2011 total costs (million)</th>
<th>June, July, Aug. 2012 total costs (million)</th>
<th>Average percentage difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP model</td>
<td></td>
<td>57.1</td>
<td>13.21</td>
<td>-</td>
</tr>
<tr>
<td>MIP/CP model</td>
<td></td>
<td>57.6 (+0.8%)</td>
<td>13.4 (+1.4%)</td>
<td>+1.1%</td>
</tr>
</tbody>
</table>

Table 4.2: Comparison of CP and MIP/CP models

4.5.2 Search phases

In OPL the programmer is able to guide search types. A first way is by using a search type other than the default, either by changing the settings in the IDE editor or by changing
a CP parameter using IBM ILOG Script. However, another way is to use search phases. Search phases define instantiation strategies that help the embedded CP Optimizer search algorithm to search more efficiently for the best solutions. While working on improving the model performance, a couple of search phases have been tested. Initially, since the model tended to give solutions containing very small intervals a search phase was defined to search for solutions with as large intervals as possible. Unfortunately though, this search phase did not improve performance as it led to solutions of only 3 intervals per line (i.e.: one interval per month, running during the whole month).

Another attempt was to make the search algorithm “prefer” intervals of medium size and highest production throughput. The least interval length was 120 hours and the maximum 720 hours (1 month). So, a search phase was created that was supposed to set the intervals length at around 420 hours. Unfortunately this search phase did not work either. Despite the efforts that were made to make it work, it was not successful.

Finally, two attempts were made with search phases that only specified the order in which variables were considered (without specifying how to choose values for these variables). In the first case, a search phase is used only for the sequence variables $seq_l, l \in L$. During this search phase, CPO decides which intervals will be present in the sequence and in which order (without setting the exact start and end times). The second search phase that we tried, was taking decisions at first for demand satisfaction variables $ds_o, o \in O$, and then for inventory level variables $I_{p,m}, p \in P$, and $m \in monthRange$. Table 4.3 presents the performance achieved by each search phase.

<table>
<thead>
<tr>
<th>Model</th>
<th>Data instance</th>
<th>Average percentage difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>April 2011</td>
<td>June 2012</td>
</tr>
<tr>
<td></td>
<td>Run time (seconds)</td>
<td>Total cost (million)</td>
</tr>
<tr>
<td>No search phase</td>
<td>247</td>
<td>43.43</td>
</tr>
<tr>
<td>Sequence search phase</td>
<td>180</td>
<td>46.03 (+6%)</td>
</tr>
<tr>
<td>$ds_o - I_{p,m}$ search phase</td>
<td>245</td>
<td>56.04 (+29%)</td>
</tr>
</tbody>
</table>

Table 4.3: Results of search phase alternatives (simple tests)

<table>
<thead>
<tr>
<th>Model</th>
<th>Data instance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>April, May, June 2011 total costs (million)</td>
</tr>
<tr>
<td>No search phase</td>
<td>57.5</td>
</tr>
<tr>
<td>Sequence search phase</td>
<td>57.1 (-0.7%)</td>
</tr>
<tr>
<td>$ds_o - I_{p,m}$ search phase</td>
<td>74.2 (+29%)</td>
</tr>
</tbody>
</table>

Table 4.4: Results of search phase alternatives

As shown in Table 4.3, on average the sequence search phase improves performance except from April 2011. However, the bigger tests show that the sequence search phase improves performance on both instances. As to the search phase for $ds_o$ and $I_{p,m}$, it deteriorates performance in all tests. Hence the sequence search phase is included to the final model. The problem of AC consists of decisions regarding the amount of production per time period in order to satisfy demands and inventory levels, in which time periods to
produce, and in what exact start and end times. Currently though, CPO cannot handle efficiently all these decisions in the same model. This can perhaps explain the improvement achieved by using the search phase of the sequence variables, since CPO first decides on which intervals will be present and in which order, and then chooses the exact start and end times.

### 4.5.3 Maximum inventory level

During testing, it was noticed that the model was building high stocks for certain products. This may not consist a violation of any constraint, but still it means that a considerable amount of production time is spent on building huge stocks that are not actually needed. So, although AC has no maximum limit for the inventory levels of products, the inventory levels of all products were limited. Two alternatives were investigated, were the limit was set to the stock target plus 120 and 200 tons respectively. Tests with limits below 120 tons (plus the stock target) are not presented. This because in practice it was shown, that in some cases the post-processing method presented in Section 4.5.8 may increase the stock even further. So for a stock level limit equal to the stock target plus 100 tons for example, there are cases where after the post processing method, this limit is exceeded and then, the problem of the next month becomes infeasible.

<table>
<thead>
<tr>
<th>Model</th>
<th>April 2011</th>
<th>June 2012</th>
<th>August 2012</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Run time</td>
<td>Total cost</td>
<td>Run time</td>
</tr>
<tr>
<td></td>
<td>(seconds)</td>
<td>(million)</td>
<td>(seconds)</td>
</tr>
<tr>
<td>No limit</td>
<td>260</td>
<td>42.92</td>
<td>95</td>
</tr>
<tr>
<td>120</td>
<td>180</td>
<td>46.03 (+7.2%)</td>
<td>103</td>
</tr>
<tr>
<td>200</td>
<td>273</td>
<td>45.64 (+6.3%)</td>
<td>94</td>
</tr>
</tbody>
</table>

Table 4.5: Maximum inventory limit results (simple tests)

For the three single month data instances, the results of the two alternatives along with the results without stock limit, are presented in Table 4.5. The results suggest that the stock limit deteriorates performance. However, when the two alternatives are tested on the bigger data instances (Table 4.6), it becomes clear that the limit of 120 tons improves performance. This is not the case for the limit of 200 tons though. Although performance is improved for April, May, June 2011 by 6.8%, for June, July, August 2012 the model performance is considerably lower resulting in an increase of 17.9% in total costs. A possible explanation could be that when the limit of 200 tons is used in the data instance of June, July, August 2012, the algorithm does not propagate so fast as in the other two cases, or that although it propagates fast, the solutions found are not as good as in the other alternatives. So, by these tests we conclude that setting an inventory limit equal to the stock target plus 120 tons performs better than the other two alternatives, and thus this modification is included in the final model configuration.

<table>
<thead>
<tr>
<th>Model</th>
<th>April, May, June 2011 total costs (million)</th>
<th>June, July, Aug. 2012 total costs (million)</th>
<th>Average percentage difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>No limit</td>
<td>57.4</td>
<td>14.5</td>
<td>-</td>
</tr>
<tr>
<td>120</td>
<td>57.1 (-0.5%)</td>
<td>13.21 (-8.9%)</td>
<td>-4.7%</td>
</tr>
<tr>
<td>200</td>
<td>53.5 (-6.8%)</td>
<td>17.1 (+17.9%)</td>
<td>+5.5%</td>
</tr>
</tbody>
</table>

Table 4.6: Maximum inventory limit results
4.5.4 Beginning of the changeover task intervals

Although the changeover task intervals are already defined accurately by Equations 4.1-4.4 of Section 4.3, adding Equation 4.23 improved the resulting solution of the model. In this equation the start time of the changeover task is set at or after the end of the previous spinning task. For this purpose, the startOf constraint is used (without the optional argument) and the endOfPrev(seq, g) constraint which returns, for sequence seq and interval g, the ending time of the interval f scheduled just before g on the same sequence seq.

For all \( i \in I^{spin}, m \in monthRange \)
\[
\text{startOf}(cot_{i,m}) \geq \text{endOfPrev}(seq_{i}, spt_{i,m}) \tag{4.23}
\]

Besides Equation 4.23, another attempt has been made to improve the performance of the constraints regarding the changeover tasks. More specifically, instead of using the startOf and endOfPrev constraints, it has been tried to use the noOverlap constraint for both spinning task and changeover task intervals. Since there is no sequence for both spinning task and changeover task intervals of the same production line, the append function has been used, which concatenates the variables. The constraint is given in Equation 4.24 below:

For all \( l \in L \)
\[
\text{noOverlap}(\text{append}(\text{all}(i \in I^{spin}, m \in monthRange | l(i) = l)spt_{i,m}, \text{all}(i \in I^{spin}, m \in monthRange | l(i) = l)cot_{i,m})) \tag{4.24}
\]

After testing the proposed additions in the three single month data instances (Table 4.7), it seems that Equation 4.23 improves performance, while Equation 4.24 does not. However Table 4.8 shows that in the bigger data instances performance is improved by both equations, but still Equation 4.23 achieves a bigger improvement. Hence, it is included in the final model configuration. Note that using both equations was not considered as an alternative, since they both model the same constraint. Hence, we decided to avoid having multiple expressions in the model, modeling the same constraint.

<table>
<thead>
<tr>
<th>Model</th>
<th>April 2011</th>
<th>June 2012</th>
<th>August 2012</th>
<th>Average percentage difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Run time (seconds)</td>
<td>Total cost (million)</td>
<td>Run time (seconds)</td>
<td>Total cost (million)</td>
</tr>
<tr>
<td>Original</td>
<td>125</td>
<td>47.15</td>
<td>93</td>
<td>9.26</td>
</tr>
<tr>
<td>Incl. Eq. 4.23</td>
<td>180</td>
<td>46.03 (-2.4%)</td>
<td>103</td>
<td>7.75 (-16.3%)</td>
</tr>
<tr>
<td>Incl. Eq. 4.24</td>
<td>227</td>
<td>43.4 (-7.9%)</td>
<td>261</td>
<td>9.76 (+5.4%)</td>
</tr>
</tbody>
</table>

Table 4.7: Results of changeover task alternatives (simple tests)

<table>
<thead>
<tr>
<th>Model</th>
<th>April, May, June 2011 total costs (million)</th>
<th>June, July, Aug. 2012 total costs (million)</th>
<th>Average percentage difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>73.7</td>
<td>15.87</td>
<td>-</td>
</tr>
<tr>
<td>Incl. Eq. 4.23</td>
<td>57.1 (-22.5%)</td>
<td>13.21 (-16.8%)</td>
<td>-19.6%</td>
</tr>
<tr>
<td>Incl. Eq. 4.24</td>
<td>66.1 (-10.3%)</td>
<td>17.42 (+9.8%)</td>
<td>-0.3%</td>
</tr>
</tbody>
</table>

Table 4.8: Results of changeover task alternatives

An alternative to Equation 4.23 was also tested and is shown in Equation 4.25. According to Equation 4.23, the model can choose to leave a gap between a spinning task and the
following changeover task (if any). Of course this makes sense if the model needs to leave a gap because of a particular constraint. However we decided to test the alternative as given in Equation 4.25, to check the effect on quality. Again the \( endOfPrev(seq_{l},spt_{i,m}) \) constraint is used, where \( l \in L, \ i \in I_{spin}, \) and \( m \in monthRange. \) The optional argument specifying the return value of \( endOfPrev \) in case \( spt_{i,m} \) is the first interval of \( seq_{l} \) is not provided and thus, the default value 0 is used. Hence if \( spt_{i,m} \) is the first interval, the start of \( cot_{i,m} \) will be set to 0, which is the beginning of the scheduling horizon.

\[
\forall i \in I_{spin}, m \in monthRange \\
startOf(cot_{i,m}) = endOfPrev(seq_{l}, spt_{i,m}) (4.25)
\]

Note that the particular constraint should not be used for data instances that include fixed tasks (like all instances of 2012). If for example there is a small fixed task in the middle of the scheduling horizon and Equation 4.25 is included in the model, then the model will have to schedule tasks either before the fixed task or after it, since if it schedules tasks in both periods, the start time of \( cot_{i,m} \) of task \( i \in I_{spin} \) scheduled just after the fixed task will not be equal to the end time of the exactly previous spinning task (which is scheduled before the fixed task) and the constraint of Equation 4.25 will be violated. Thus this alternative is only tested on the instance of April, May, June 2011, where no fixed tasks are included. Initially Equation 4.25 was introduced to avoid any unnecessary gaps in the schedule. However, after analyzing carefully the schedules with and without Equation 4.25, we found that the model leaves no gaps in the schedule anyway.

Table 4.9 presents the results, indicating that there is an improvement in the solution quality. Only the inventory deficit costs have a larger value, but still the total costs are lowered by 5.4%. Table 4.10 presents a comparison of Equations 4.23 and 4.25, when testing them on the first month of the April, May, June 2011 data instance (i.e., data instance April 2011) with exactly the same configurations as in Table 4.9, for 100000, 200000, and 500000 fails. As shown, for the 100000 fails when Equation 4.23 is included, the model requires 7.5 seconds less and gives just a little better results than the model with Equation 4.25. However, when the fail limit is doubled, the model with Equation 4.25 requires considerably less time and it gives a solution of 2.37 million less. The same trend continues for a fail limit of 500000. These results justify the performance improvement achieved by Equation 4.25 on Table 4.9, as it seems to perform better than Equation 4.23 for longer running times.

<table>
<thead>
<tr>
<th>Model</th>
<th>Non delivery costs (million)</th>
<th>Inventory deficit costs (million)</th>
<th>Setup duration (hours)</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eq. 4.23</td>
<td>41</td>
<td>18.4</td>
<td>1030</td>
<td>0.94</td>
<td>60.3</td>
</tr>
<tr>
<td>Eq. 4.25</td>
<td>35.3</td>
<td>20.9</td>
<td>850</td>
<td>0.8</td>
<td>57.1 (-5.4%)</td>
</tr>
</tbody>
</table>

Table 4.9: Performance comparison of Equations 4.23 and 4.25 on April, May, June 2011

<table>
<thead>
<tr>
<th>Model</th>
<th>( 10^b ) fails</th>
<th>( 2 \times 10^b ) fails</th>
<th>( 5 \times 10^b ) fails</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Run time (seconds)</td>
<td>Total costs (million)</td>
<td>Run time (seconds)</td>
</tr>
<tr>
<td>Eq. 4.23</td>
<td>164.1</td>
<td>45.8</td>
<td>498.5</td>
</tr>
<tr>
<td>Eq. 4.25</td>
<td>179.5</td>
<td>46.03</td>
<td>375.75</td>
</tr>
</tbody>
</table>

Table 4.10: Performance comparison of Equations 4.23 and 4.25 on April 2011
4.5.5 Initial changeovers

Although Equation 4.12 is the most straightforward way of modeling the constraint for initial changeovers, an alternative has been tried. Instead of setting the start time of all intervals to be after the changeover time that would be needed between the initial recipes of the production lines and the recipes of these intervals (if they were scheduled first), only the intervals whose start time is less than \( m_{pt} \) are considered, using meta-constraints. Equation 4.26 below consists of two parts, where the first part is used to check whether the start time of the interval is less than \( m_{pt} \) or not. If this is true, then by using the => symbol the next statement must hold which states that the start time of this particular interval must be at least equal to the changeover time between the initial recipe of the line and the recipe of the interval.

For all \( i \in I_{spin}, \ m \in monthRange \)

\[
(\text{startOf}(spt_{i,m},H) < m_{pt}) \Rightarrow (\text{startOf}(spt_{i,m}) \geq \text{et}(\text{curRec}_{(l)}, r(i)))
\]  

\[(4.26)\]

<table>
<thead>
<tr>
<th>Model</th>
<th>April 2011</th>
<th>June 2012</th>
<th>August 2012</th>
<th>Average percentage difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Run time (seconds)</td>
<td>Total cost (million)</td>
<td>Run time (seconds)</td>
<td>Total cost (million)</td>
</tr>
<tr>
<td>Eq. 4.12</td>
<td>277</td>
<td>48.76</td>
<td>81</td>
<td>8.5</td>
</tr>
<tr>
<td>Eq. 4.26</td>
<td>180</td>
<td>46.03 (-5.6%)</td>
<td>103</td>
<td>7.75 (-8.8%)</td>
</tr>
</tbody>
</table>

Table 4.11: Performance comparison of Equations 4.12 and 4.26 (simple tests)

As shown in Tables 4.11 and 4.12, despite the use of meta-constraints in Equation 4.26 the model performance is improved. Also, especially in the case of April 2011, the time required for the 100000 fails is far less when Equation 4.26. The reason for this improvement is probably owed to the fact that the meta-constraint does not consider all present intervals, but only the ones that start before \( m_{pt} \). Since the length of a task cannot be less than \( m_{pt} \) (actually less than 156 hours according to Section 4.5.7), it means that for this constraint at most 11 intervals are considered (equal to the number of production lines). In contrast, Equation 4.12 needs to consider all present intervals and thus it is less efficient. So, Equation 4.26 is included in the configuration of the final model.

<table>
<thead>
<tr>
<th>Model</th>
<th>April, May, June 2011 total costs (million)</th>
<th>June, July, Aug. 2012 total costs (million)</th>
<th>Average percentage difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using Eq. 4.12</td>
<td>57.5</td>
<td>15.8</td>
<td>-</td>
</tr>
<tr>
<td>Using Eq. 4.26</td>
<td>57.1 (-0.6%)</td>
<td>13.21 (-16.4%)</td>
<td>-8.5%</td>
</tr>
</tbody>
</table>

Table 4.12: Performance comparison of Equations 4.12 and 4.26

4.5.6 Search space reduction

As mentioned in Section 4.5.1, the problem of AC is too large. Hence several constraints are introduced to reduce the search space of the problem. That way we expect that the model will reach better solutions faster.

**One interval per line per product:** As explained, during a particular month \( m \) and for each task \( i \in I_{spin} \), a single interval \( spt_{i,m} \) is created. This means that during a month, for each production line and product, a recipe can be used only once. This, in combination
with the fact that the model had the tendency to schedule the intervals to finish as soon as possible, resulted in using several different recipes during one month on a single line for the production of a single product. Besides the time lost for changeovers, recipes of lower production throughput were used degrading the model performance. To tackle this problem, during any month the model is forced to choose a single recipe per production line for producing a particular product. This is modeled with the use of a new set of intervals named \( productInt_{l,p,m} \) where \( l \in L, p \in P, \) and \( m \in monthRange, \) and the alternative constraint. For each line \( l \) and month \( m, \) if the model chooses to produce a product \( p, \) it can do so by choosing a single recipe only. The constraint is presented in Equation 4.27 below:

For all \( l \in L, \ p \in P, \ m \in monthRange \)

\[
\text{alternative}(productInt_{l,p,m}, all(i \in I_{spin} | l(i) = l \land pr_r(i) = p) spt_{i,m})
\] (4.27)

**Produce each product at most 4 times per month:** Another idea for decreasing the search space of the model was to limit the maximum number of times that a product can be scheduled for production on all lines during a single month. By analyzing the results of the model, it was found that each product was not scheduled more than 4 times per month. So, by setting a limit of 4, the search space would be reduced considerably, while the solutions obtained before would still be feasible. This constraint is modeled as shown in Equation 4.28.

For all \( p \in P, \ m \in monthRange \)

\[
\sum_{i \in I_{spin}} \text{presenceOf}(spt_{i,m}) \leq 4
\] (4.28)

**At most 5 spinning task intervals per production line per month:** A final idea that was tested for the reduction of the search space, was to limit the number of present intervals on each production line and each month to 5. Again, the limit of 5 was chosen, since the model solutions for a given month did not contain more intervals on the same production line. Equation 4.29 shows how this is modeled:

For all \( l \in L, \ m \in monthRange \)

\[
\sum_{i \in I_{spin}} \text{presenceOf}(spt_{i,m}) \leq 5
\] (4.29)

<table>
<thead>
<tr>
<th>Model</th>
<th>April 2011</th>
<th>June 2012</th>
<th>August 2012</th>
<th>Average percentage difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Run time (seconds)</td>
<td>Total cost (million)</td>
<td>Run time (seconds)</td>
<td>Total cost (million)</td>
</tr>
<tr>
<td>None</td>
<td>258</td>
<td>39.69</td>
<td>90</td>
<td>11.62</td>
</tr>
<tr>
<td>Incl. Eq. 4.27</td>
<td>180</td>
<td>46.03 (+15.9%)</td>
<td>103</td>
<td>7.75 (-33.3%)</td>
</tr>
<tr>
<td>Incl. Eq. 4.28</td>
<td>220</td>
<td>38.66 (-2.8%)</td>
<td>79</td>
<td>11.08 (-4.6%)</td>
</tr>
<tr>
<td>Incl. Eq. 4.29</td>
<td>225</td>
<td>40.07 (+1%)</td>
<td>90</td>
<td>7.9 (-32%)</td>
</tr>
</tbody>
</table>

Table 4.13: Results of search space reduction constraints (simple tests)

The effect of Equations 4.27, 4.28, and 4.29 on the model without any search space reduction equations is given in Table 4.13. As shown, on average, all equations seem to improve more or less the model performance. However, in order to avoid multiple productions of the same product on the same production line during a particular month,
we decided to include Equation 4.27 anyway. Thus, in April, May, June 2011 and June, July, August 2012 only combinations of Equation 4.27 with Equations 4.28 and 4.29 are tested and presented in Table 4.14.

<table>
<thead>
<tr>
<th>Model</th>
<th>Data instance</th>
<th>April, May, June 2011</th>
<th>June, July, Aug. 2012</th>
<th>Average percentage difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>total costs</td>
<td>total costs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(million)</td>
<td>(million)</td>
<td></td>
</tr>
<tr>
<td>Including Eq. 4.27</td>
<td></td>
<td>57.1</td>
<td>13.21</td>
<td>-</td>
</tr>
<tr>
<td>Incl. both Eq. 4.27 and 4.28</td>
<td></td>
<td>58.8 (+3%)</td>
<td>14.4 (+9%)</td>
<td>+6%</td>
</tr>
<tr>
<td>Incl. both Eq. 4.27 and 4.29</td>
<td></td>
<td>55 (-3.6%)</td>
<td>17.3 (+31%)</td>
<td>+13.7%</td>
</tr>
<tr>
<td>Including all Equations</td>
<td></td>
<td>48.71 (-15.9%)</td>
<td>12.89 (-2.4%)</td>
<td>-9.1%</td>
</tr>
</tbody>
</table>

Table 4.14: Results of search space reduction constraints

From the results it becomes clear that using all three equations for the reduction of the search space, gives the best results. Although the combination of Equations 4.27 and 4.28 or the combination of 4.27 and 4.29 indicate that performance is deteriorated, when combining all three equations, the results are improved. This model behavior can be explained in a similar way as explained in Section 4.5.3.

4.5.7 Minimum processing time

In the plant of AC there is a minimum allowed processing time \((mpt)\) of 120 hours for each recipe \(r \in R\). Unfortunately the CPO engine tends to schedule the intervals to finish as soon as possible. This means that CPO tends to give intervals the minimum possible length, resulting in a solution with many small intervals. So, we decided to investigate whether raising \(mpt\) to 132, 144, and 156 hours would have a beneficial effect on the solution quality. Tests with values higher than 156 hours were not done, since we decided not to deviate that much from the particular non-technical constraint set by AC. Since Section 4.5.6 shows that the combination of Equations 4.27, 4.28, and 4.29 gives better results, Equations 4.28 and 4.29 are also included in the tests of the 3-months data instances. This because the increase of \(mpt\) constitutes a constraint violation. Thus we performed the tests in the best possible model to have a clearer view of the performance effect and whether this violation is really worth it.

The results of Table 4.15 suggest that the increase of \(mpt\) improves performance, except from the case where \(mpt = 132\). In that case, although the results are worse, the model requires less time to reach the limit of 100000 fails, than when \(mpt = 120\). So, it is not certain that in the bigger tests the results will be worse too. Indeed, in the 3-months data instances (Table 4.16), when \(mpt = 132\) the results are better than when \(mpt = 120\). As shown in both tables, the best results are achieved on average when \(mpt = 156\). The improvement is considerable (25.2%), so we decided to deviate from the non-technical constraint of AC and increase the minimum allowed processing time from 120 to 156 hours. As explained in Section 4.5.8, during the post processing method \(mpt\) is set back to 120 hours, so that the model can fill any gaps with smaller productions as well.

4.5.8 Post processing

A final performance improvement was achieved by introducing a post processing method, consisting of three steps, running after each iteration of the decomposition model. The model of the post processing method is quite similar to the decomposition model. This model contains the same constraints and loads the same data as the decomposition model. The difference is that the solution of the decomposition model (i.e.: inventory level of
Table 4.15: Results of different mpt values (simple tests)

<table>
<thead>
<tr>
<th>mpt (hours)</th>
<th>Average percentage difference</th>
<th>April 2011</th>
<th>June 2012</th>
<th>August 2012</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Run time</td>
<td>Total cost</td>
<td>Run time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(seconds)</td>
<td>(million)</td>
<td>(seconds)</td>
</tr>
<tr>
<td>120</td>
<td>-</td>
<td>120</td>
<td>245</td>
<td>10</td>
</tr>
<tr>
<td>132</td>
<td>+7.2%</td>
<td>132</td>
<td>163</td>
<td>84</td>
</tr>
<tr>
<td>144</td>
<td>-1.1%</td>
<td>144</td>
<td>234</td>
<td>88</td>
</tr>
<tr>
<td>156</td>
<td>-9.5%</td>
<td>156</td>
<td>180</td>
<td>103</td>
</tr>
</tbody>
</table>

Table 4.16: Results of different mpt values

<table>
<thead>
<tr>
<th>mpt (hours)</th>
<th>Data instance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>April, May, June 2011 total costs (million)</td>
</tr>
<tr>
<td>120</td>
<td>57.2</td>
</tr>
<tr>
<td>132</td>
<td>55.8 (-2.5%)</td>
</tr>
<tr>
<td>144</td>
<td>53.4 (-6.6%)</td>
</tr>
<tr>
<td>156</td>
<td>48.71 (-14.8%)</td>
</tr>
</tbody>
</table>

Each product, satisfied amount of each production order, and the actual spinning task and changeover task intervals) is imported as well. The availableLength\(_{i,m}\) of \(spt_{i,m}\), where \(i \in I^{\text{spin}}, m \in \text{monthRange}\), is also calculated for all present intervals and it is equal to the free time between the end of interval \(spt_{i,m}\) and the beginning of the following changeover task interval \(cot_{i,m}\). In other words, availableLength\(_{i,m}\) is the gap between two consecutive intervals. Another difference is that the value of mpt is set back to 120 hours, while the limit on inventory levels is removed. Hence the post processing method will be able to fill gaps of at least 5 days, either by adding new tasks, or by stretching the already existing ones. Note that in all three steps, the inventory levels and the satisfied demand amounts (as specified by the decomposition model or the previous post processing step) are used as lower limits for the corresponding variables. That way it is ensured that the solution of each step will be at least as good as the solution obtained by the decomposition model or the previous post processing step.

First step: The purpose of the first step is to inflate productions as well as add new ones, in order to fill any gaps in the schedule and improve the objectives. During this step, the model has to make present all intervals that were present in the solution of the decomposition model, and it can set their start time to be at most equal to the start time they had in the solution of the decomposition model. As to the length of the tasks, it must be at least equal to the length they had before. Obviously, the model is able to add new intervals as well. By using the sum of gaps of all present intervals (i.e., the sum of availableLength\(_{i,m}\)) in the objective function, any intervals that are not as long as they could be, are stretched to fill any idling times of the schedule. Besides the sum of gaps over all intervals, the objective function consists again of the inventory deficit cost and the non delivery cost, where the latter has a lower weight factor. That way the model concentrates more on filling the gaps and minimizing the inventory deficit costs.

Second step: As to the second post processing step, its main purpose is to reduce the changeover costs. To do so, the model can now change the recipes of the intervals. More specifically, although the model has to schedule productions of the same products on the same production lines, it can now choose a different recipe. In the solution of the decom-
position model, sometimes it was noticed that the model made some weird decisions where it would schedule an unnecessary die changeover. For example, although the same product can be produced with a different recipe of the same throughput without needing a different die, the model may choose the recipe requiring the die changeover for no particular reason. Thus this step is introduced to avoid such changeovers. By freeing production time, new intervals can be scheduled or the already existing ones can be stretched further, resulting in a further improvement of the objectives. For this step, $O_{sc}$ has obviously a big weight factor in the objective function, making sure that the minimization of changeovers is a priority.

**Third step:** Finally, the third step takes over, whose main purpose is again to reduce the changeover costs. In this step, the model has to schedule again all intervals that were present in the second step. Also, all intervals have to be at least as long as they were before. The difference is that now the model can change the sequence of the intervals (within the same month), and thus reduce changeovers. If the decomposition model makes a weird decision of scheduling an unnecessary sequence of die changeovers, this post processing step will correct it. For instance, if the decomposition model chooses on a particular production line to use die K, then L and then K again for no particular reason, this step will try to change the sequence of the intervals and will use the same dies either as K-K-L or as L-K-K. Again, by reducing the changeover time, production time is freed that can be used to stretch any useful productions or add new ones. The objective function is the same as in the second step where the changeover costs are a priority.

The performance improvement achieved by these steps is presented in Table 4.17. The table shows how the performance is improved further after adding each step of the proposed post processing method. However, a step may not manage to improve performance. This is shown for example in the data instance of April 2011, where there is no difference in total costs between the first and the second post processing step. Note that all percentages indicate the difference if compared to the solution without post processing. Also note that for all tests, the running time of the decomposition model was 3000 seconds, while each post processing step ran for 100000 fails. Finally, all modifications that contributed in the performance improvement are included in the tests.

Table 4.17 cannot show the effect on changeovers, since it only shows the total costs for each data instance. Table 4.18 shows how the total changeover duration changes by introducing the post processing steps. After the first step, changeovers are increased. This was expected, since the main task of the first step is to “inflate” current productions as well as add new ones. Thus more changeovers are added to the schedule. However, when the next two steps are executed, the total duration of changeovers drops. Table 4.17 gives also the running time of each step as a reference. As shown, the time required by each step is not long, with the exception of the third step of August 2012.

Finally, steps 2 and 3 could be of course combined in a single step, where the model would be able to change the recipes as well as the sequence of the intervals. Initially
|
|---|---|---|
|Post processing steps | April 2011 changeover duration (hours) | June 2012 changeover duration (hours) | August 2012 changeover duration (hours) |
|No post processing | 164 | 224 | 389 |
|After 1st step | 353 | 272 | 389 |
|After 2nd step | 298 | 217 | 246 |
|After 3rd step | 283 | 209 | 235 |

Table 4.18: Changeover duration difference after each post processing step

this alternative has been tried, but unfortunately the model did not manage to find any solution in satisfyingly short running time. So, in order to avoid having a post processing method that required to run for too long, two separate steps were created. Due to time limitations, a comparison between the two alternatives has not been performed.

### 4.6 Introducing Fixed Productions, Maintenance Stops and Production Line Shut Downs

In the practice of AC it often occurs that some fixed productions need to be scheduled on a particular production line. These productions correspond to a production of a product using a regular recipe or even a new “experimental” one. In the case of experimental recipes, the amount produced is not added to the inventories and also it does not satisfy any demand. Furthermore, maintenance stops are scheduled regularly to ensure the smooth operation of the production facility. Such stops concern a particular production line which is not operational during the maintenance period. Finally, in light of the decreased product demand, AC decided to shut down some production lines to reduce operational costs. Obviously such decisions must be taken under consideration by the model as well.

A database with a table containing all fixed productions (including the amounts produced by any regular recipes), maintenance stops, and shut downs is provided as input to the model. Then the model filters the entries and considers the ones overlapping with the month that is currently solving. The start and end times of the entries are trimmed so that they do not extend outside the planning horizon. Afterwards an interval is created for each of these fixed tasks. Let $FT$ denote the set of fixed tasks. Then the interval variables $fixedInterval_k$, $k \in FT$ are created. Since such intervals occupy the production line where they are scheduled, the noOverlap constraint is used as shown in Equation 4.30 below:

\[
\text{noOverlap} (\text{append} \ (all(i \in I^{\text{spin}}, \ m \in \text{monthRange} | \ l(i) = l) \ spt_{i,m}, \ all(k \in FT | l(k) = l) \ fixedInterval_k))
\] (4.30)

Another way of modeling the fixed tasks, is by creating a stepwise linear function $fixed_l$ for each production line $l \in L$, indicating the start and end times of the corresponding fixed tasks. Then by using the forbidExtent constraint as shown in Equation 4.31, it is ensured that the spinning tasks do not overlap with any fixed task.

\[
\text{forbidExtent} (spt_{i,m}, fixed_{l(i)})
\] (4.31)

Between the two different ways of modeling, the use of the stepwise linear functions gives better results. This is shown in Table 4.19 where the performance of both modeling
methods is tested on the data instance of June, July, August 2012, with the same parameters as described in Section 4.5. The only drawback of Equation 4.31 is that it gives a solution of 4 more changeovers and thus, the changeover cost is increased by 40000 Euro. However, although there are more changeovers, the total changeover duration is less by 51 hours.

<table>
<thead>
<tr>
<th></th>
<th>Non delivery costs (million)</th>
<th>Inventory deficit costs (million)</th>
<th>Setup duration (hours)</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eq. 4.30</td>
<td>3.6</td>
<td>11.2</td>
<td>698</td>
<td>0.6</td>
<td>15.5</td>
</tr>
<tr>
<td>Eq. 4.31</td>
<td>2.37</td>
<td>9.82</td>
<td>647</td>
<td>0.7</td>
<td>12.89 (-16.8%)</td>
</tr>
</tbody>
</table>

Table 4.19: Performance comparison of Equations 4.30 and 4.31 on June, July, August 2012

### 4.7 Final Model Configuration

After completing the testing presented above, we came up with the final model configuration that gives the best results. This includes the following:

- Size decomposition is applied (Section 4.5.1)
- The sequence search phase is used (Section 4.5.2)
- A maximum inventory level is set, equal to 120 tons per product (Section 4.5.3)
- Equation 4.25 is included for the data instance of April, May, June 2011, while Equation 4.23 is included for the data instance of June, July, August 2012 (Section 4.5.4)
- Equation 4.12 is replaced by Equation 4.26 (Section 4.5.5)
- The search space is reduced using Equations 4.27, 4.28 and 4.29 (Section 4.5.6)
- The minimum processing time \(mpt\) is raised from 120 to 156 hours (Section 4.5.7)
- The post processing method is used (Section 4.5.8)
- For the data instance of June, July, August 2012 where fixed tasks are present, Equation 4.31 is used (Section 4.6)

Note that the particular configuration is used in all tests of this report, unless it is stated otherwise.
Chapter 5

Evaluation of Constraint Programming Model

In this chapter a wide evaluation of the CP model is performed, to investigate its behavior and performance under different cases. First, in Section 5.1 it is explained how the programmer can limit the search of the model and also how the obtained results are deterministic. Then, various evaluation methods that have been used to verify the correctness of the model are presented in Section 5.2. In total six different cases were tested, which are identical to the ones used in [29]. The purpose of these cases is to check how the model behaves under certain different scenarios. For each scenario, the solution proposed by the model is presented, along with a comparison to the results of the model developed in [29]. Section 5.3 presents how different model running times impact the end result of the model. Similarly, Section 5.4 shows how the different weights of the objective function affect the solution quality of the model, while in Section 5.5 tests are performed using alternative objectives (other than the ones used in $OF$), in an attempt to approach the problem from a different point of view. Next, Section 5.6 shows the influence of different algorithm seed values to the model behavior, and then Section 5.7 follows where the effect of relaxing particular constraints is investigated. Finally Section 5.8 gives the results of a model run on a larger data instance, and Section 5.9 compares a model solution to the corresponding manual plan, in an attempt to understand whether the model can be used in practice.

5.1 Search Limit and Solution Determinism

As said, one of the main drawbacks of the solution proposed in [29] is that it was not deterministic, and because of that it was considered to not be usable in practice. In other words, a different solution would be given, each time the model was run. To deal with this problem we developed a deterministic model.

However, the developed CP model is too big for the CP Optimizer to find the optimal solution. In such cases, OPL offers two alternatives for limiting the search. In the first way a limit can be set to the running time of the model, while in the second way the programmer can set a limit on the number of fails made by the search algorithm. Obviously though, both ways restrict the model from finding the optimal solution, unless it is found earlier.

When a running time limit is set, the search will run for the allowed time and it will return the best solution found so far (unless it finds the optimal solution earlier). Hence the performance of the model depends strongly on the computer processor and its load. For instance, if the processor is busy with other demanding tasks as well, it is probable that the resulting solution will be worse. Note though that the particular variability is not similar to what is described in [29]. The algorithm always performs the same and it is guaranteed to give the same result in multiple runs of the same instance, provided that
the running time is exactly equal and that the computer processor has exactly the same load.

As to the fail limit, it is certain that for a given data instance, the model will always return the same solution. As to the time required, the search will also spend the same time when it is run on the same hardware configuration under the same load. The only negative aspect of using fail limits, is that for two different data instances, the search does not require the same amount of time to reach the particular fail limit. In other words, when search is limited to X fails, it may take more time in one data instance than in the other. Thus, it cannot be known beforehand, whether a particular fail limit will take too long or too short for a given data instance.

From the above, it is made clear that the model is capable of giving deterministic results when a fail limit is used, irrespective of the hardware configuration or the current load of the CPU. In this chapter, all tests are performed using time limit, since the model was run on the same hardware configuration, while making sure that no other demanding tasks were run on the CPU. In any case, when the model is run, COS provides the number of fails made by the search. So, even if the search is limited using a time limit, when the results need to be reproduced, the user may limit the search of each month by the number of fails reached by the time-limited search of each month.

5.2 Validation Cases

The information provided in these validation cases is considered classified. Hence they are not presented in this version of the document.

5.3 Effect of Running Time Limit

Since the problem of AC is too large for the CP model to find the optimal solution, a search limit has to be set. This section investigates how the increase of the time limit affects the solution quality of the single month data instance April 2011, and then of the three months data instances April, May, June 2011 and June, July, August 2012. Note that in all these tests, the post processing method is also included with a fail limit of 100000 fails per step. The post processing method is included in these tests, as we wanted to investigate the effect of the running time to the complete model. As to the weight factors of OF, the values are \( W_{ndc} : W_{idc} : W_{sc} = 100 : 1 : 0.05 \), where \( W_{ndc} \) is the weight of non delivery cost and \( W_{idc}, W_{sc} \) are the weights of inventory deficit and setup costs respectively. Figure 5.1 illustrates the development of total cost as a function of running time, for the data instance of April 2011. As shown in the graph, the total cost as a function of time, does not have a constant improvement rate but it varies. For instance, the improvement from 1000 seconds to 2000 seconds is 5.2 million, while the improvement from 4000 seconds to 5000 seconds is just 0.5 million. Furthermore the graph shows that the most improvement is achieved during the first 3000 to 4000 seconds and afterwards the cost decrease rate is reduced. The numbers of Figure 5.1 are also presented in Table 5.1.

<table>
<thead>
<tr>
<th>Run time (seconds)</th>
<th>30</th>
<th>500</th>
<th>10³</th>
<th>2 · 10³</th>
<th>3 · 10³</th>
<th>3.5 · 10³</th>
<th>4 · 10³</th>
<th>5 · 10³</th>
<th>6 · 10³</th>
<th>7 · 10³</th>
<th>10⁴</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total costs (million)</td>
<td>49.9</td>
<td>30.9</td>
<td>30</td>
<td>24.8</td>
<td>23.3</td>
<td>20.3</td>
<td>19.8</td>
<td>19.3</td>
<td>19</td>
<td>17.8</td>
<td>17.6</td>
</tr>
</tbody>
</table>

Table 5.1: The total cost of April 2011 for various running time limits

Tables 5.2 and 5.3 present the results of different running times for the two three
Figure 5.1: The total cost of April 2011 for various running time limits

months data instances. Note that all limits are expressed in seconds and that they represent the time limit per month.

<table>
<thead>
<tr>
<th>Run time limit (seconds)</th>
<th>Non delivery costs (million)</th>
<th>Inventory deficit costs (million)</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>48.94</td>
<td>17.47</td>
<td>0.89</td>
<td>67.3</td>
</tr>
<tr>
<td>2000</td>
<td>28.81</td>
<td>19.34</td>
<td>0.82</td>
<td>48.97</td>
</tr>
<tr>
<td>3000</td>
<td>29.01</td>
<td>18.93</td>
<td>0.77</td>
<td>48.71</td>
</tr>
<tr>
<td>3500</td>
<td>28.33</td>
<td>21.86</td>
<td>0.71</td>
<td>50.9</td>
</tr>
<tr>
<td>4000</td>
<td>21.99</td>
<td>19.02</td>
<td>0.66</td>
<td>41.67</td>
</tr>
</tbody>
</table>

Table 5.2: Results of April, May, June 2011 for different run time limits

As expected, the solution quality of June, July, August 2012 improves when the run time limit is increased. This is not the case though for the data instance of April, May, June 2011, where the solution for 3000 seconds is better than the solution for 3500 seconds. To analyze this behavior the results are examined in a deeper level as shown in Tables 5.4 and 5.5.

Table 5.3: Results of June, July, August 2012 for different run time limits

As shown in the detailed results, for month April when the limit is set to 3500 seconds, the model performs better by almost 3 million euros in total. This is achieved by a big reduction in non delivery costs for the particular month. In the 3500 seconds run, the fully satisfied demands are 28 and the total delivered amount is 1760 tons, while in the 3000 seconds run there are 32 fully satisfied demands and a total of 1792 tons is delivered. This indicates that the model achieves better non delivery costs by choosing more correctly what amounts to deliver for each demand. However, there is a big increase in inventory deficit costs for the 3500 seconds run. The products below stock target are 38 and the
<table>
<thead>
<tr>
<th>Month of data instance</th>
<th>Non delivery costs (million)</th>
<th>Inventory deficit costs (million)</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>April</td>
<td>17.56</td>
<td>5.53</td>
<td>0.26</td>
<td>23.35</td>
</tr>
<tr>
<td>May</td>
<td>6.63</td>
<td>7.22</td>
<td>0.23</td>
<td>14.08</td>
</tr>
<tr>
<td>June</td>
<td>4.82</td>
<td>6.18</td>
<td>0.28</td>
<td>11.28</td>
</tr>
<tr>
<td>Total</td>
<td>29.01</td>
<td>18.93</td>
<td>0.77</td>
<td>48.71</td>
</tr>
</tbody>
</table>

Table 5.4: Detailed results of April, May, June 2011 for time limit of 3000 seconds per month

<table>
<thead>
<tr>
<th>Month of data instance</th>
<th>Non delivery costs (million)</th>
<th>Inventory deficit costs (million)</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>April</td>
<td>12.36</td>
<td>7.81</td>
<td>0.14</td>
<td>20.31</td>
</tr>
<tr>
<td>May</td>
<td>8.72</td>
<td>7.29</td>
<td>0.27</td>
<td>16.28</td>
</tr>
<tr>
<td>June</td>
<td>7.25</td>
<td>6.76</td>
<td>0.3</td>
<td>14.31</td>
</tr>
<tr>
<td>Total</td>
<td>28.33</td>
<td>21.86</td>
<td>0.71</td>
<td>50.9</td>
</tr>
</tbody>
</table>

Table 5.5: Detailed results of April, May, June 2011 for time limit of 3500 seconds per month

number of products with a stock level of more than 40 tons above the stock target is 4 for the 3500 seconds run. In contrast, for the 3000 seconds run these numbers are 31 and 0 respectively. This indicates that in the 3500 seconds run, the model does not use the production time to build stocks for the products that actually need it, but it builds bigger stocks that are not serving any particular reason.

The results of the next two months show that the high stock levels in the first month of the 3500 seconds run is quite detrimental for the development of the non delivery costs. For instance, the number of fully satisfied demands of the second month is 25 for the 3500 seconds run and 40 for the 3000 seconds run. This justifies the big difference in the non delivery costs of the two runs at the second month.

Hence, it becomes clear that finding the optimal solution (or more generally a better solution) for one month, may result in far worse solutions for the upcoming months, because of the particular decisions that are taken by the model. This is a disadvantage of the size decomposition that is applied to this model. Since the model does not consider the data of the following months, it may give a solution that although it is better for the current month, it creates such a starting point for the next month, that it is very hard for the model to deliver good results in total.

### 5.4 Effect of Objective Weights

The weights used in the objective function $OF$ can have a big impact on the behavior of the model and the quality of the solutions. Thus several tests were performed on both April, May, June 2011 and June, July, August 2012 data instances. For all tests, the running time of the model was limited to 3000 seconds per month, while each post processing step was run for 100,000 fails. Table 5.6 presents the results of the model using various objective weights for the data instance of April, May, June 2011, while the results for June, July, August 2012 are given in Table 5.7. The effect of the various weights is also illustrated in Figures 5.2 and 5.3. In all tables and figures, the weights are given in the form $W_{ndc} : W_{idc} : W_{sc}$.

As illustrated by the results, the model is not capable of handling efficiently cases where $W_{ndc}$ and $W_{idc}$ have similar values. When the non-delivery cost gets bigger weights
though, the solution quality is improved considerably. Obviously in these cases inventory
deficit costs are increased, but this increase is very small if compared to the decrease of
non-delivery costs. In contrast, in cases where the $W_{ndc}:W_{idc}$ ratio is reversed (i.e., when
$W_{idc}$ gets far bigger weights than $W_{ndc}$), the total costs are not of similar quality. This was
expected, since the non satisfaction of demands is more costly than the non satisfaction
of stock targets. So the lowest total costs are obtained when $W_{ndc}$ is much higher than
$W_{idc}$, and more specifically when $W_{ndc}:W_{idc}=100:0.1$.

Note that in all tests, $O_{sc}$ is part of the objective function with a weight of $W_{sc}=0.05$. Obviously, since $O_{sc}$ gets values within the range $[150, 400]$ approximately, multiplying it with a weight of 0.05 results in a negligible amount in $OF$, if compared to $O_{ndc}$ and $O_{idc}$ which represent millions. However, during the model development and during testing, the presence of $O_{sc}$ in the objective function (even with a small weight) appeared to be important to the model performance. Table 5.8 gives the results of the model when $W_{sc}$ is set to 0, while $W_{ndc}:W_{idc}=100:1$. The model was again tested on April, May, June 2011 and June, July, August 2012 with a time limit of 3000 seconds per month and a fail limit of 100000 fails per post processing step. As it is indicated, the model performs a lot worse than in the case where $W_{ndc}:W_{idc}:W_{sc}=100:1:0.05$. 

### Table 5.6: Results of April, May, June 2011 for different objective weights

<table>
<thead>
<tr>
<th>Weights</th>
<th>Non delivery costs (million)</th>
<th>Inventory deficit costs (million)</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 : 100 : 0.05</td>
<td>74.5</td>
<td>0.2</td>
<td>0.87</td>
<td>75.5</td>
</tr>
<tr>
<td>1 : 2 : 0.05</td>
<td>87.1</td>
<td>15.7</td>
<td>1.05</td>
<td>103.9</td>
</tr>
<tr>
<td>1 : 1 : 0.05</td>
<td>82.2</td>
<td>18</td>
<td>1.12</td>
<td>101.3</td>
</tr>
<tr>
<td>2 : 1 : 0.05</td>
<td>51.4</td>
<td>20.5</td>
<td>1.07</td>
<td>73</td>
</tr>
<tr>
<td>5 : 1 : 0.05</td>
<td>63.5</td>
<td>20.9</td>
<td>0.99</td>
<td>85.4</td>
</tr>
<tr>
<td>25 : 1 : 0.05</td>
<td>49</td>
<td>18.7</td>
<td>1.01</td>
<td>68.8</td>
</tr>
<tr>
<td>50 : 1 : 0.05</td>
<td>51.6</td>
<td>18.6</td>
<td>1.01</td>
<td>71.1</td>
</tr>
<tr>
<td>100 : 1 : 0.05</td>
<td>29.01</td>
<td>18.93</td>
<td>0.77</td>
<td>48.71</td>
</tr>
<tr>
<td>100 : 0.5 : 0.05</td>
<td>19.8</td>
<td>22.7</td>
<td>0.77</td>
<td>43.3</td>
</tr>
<tr>
<td>100 : 0.1 : 0.05</td>
<td>12.1</td>
<td>26.3</td>
<td>0.65</td>
<td>39.1</td>
</tr>
</tbody>
</table>

### Table 5.7: Results of June, July, August 2012 for different objective weights

<table>
<thead>
<tr>
<th>Weights</th>
<th>Non delivery costs (million)</th>
<th>Inventory deficit costs (million)</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 : 100 : 0.05</td>
<td>28.2</td>
<td>0.16</td>
<td>0.63</td>
<td>29</td>
</tr>
<tr>
<td>1 : 2 : 0.05</td>
<td>44.7</td>
<td>10.8</td>
<td>0.82</td>
<td>56.3</td>
</tr>
<tr>
<td>1 : 1 : 0.05</td>
<td>26.2</td>
<td>12</td>
<td>0.85</td>
<td>39.1</td>
</tr>
<tr>
<td>2 : 1 : 0.05</td>
<td>34.5</td>
<td>13.2</td>
<td>0.84</td>
<td>48.5</td>
</tr>
<tr>
<td>5 : 1 : 0.05</td>
<td>17.3</td>
<td>13</td>
<td>0.71</td>
<td>31</td>
</tr>
<tr>
<td>25 : 1 : 0.05</td>
<td>6.6</td>
<td>14.5</td>
<td>0.69</td>
<td>21.8</td>
</tr>
<tr>
<td>50 : 1 : 0.05</td>
<td>6.4</td>
<td>14.3</td>
<td>0.71</td>
<td>21.4</td>
</tr>
<tr>
<td>100 : 1 : 0.05</td>
<td>2.37</td>
<td>9.82</td>
<td>0.7</td>
<td>12.89</td>
</tr>
<tr>
<td>100 : 0.5 : 0.05</td>
<td>0.39</td>
<td>9.48</td>
<td>0.65</td>
<td>10.52</td>
</tr>
<tr>
<td>100 : 0.1 : 0.05</td>
<td>0.19</td>
<td>9.4</td>
<td>0.73</td>
<td>10.3</td>
</tr>
</tbody>
</table>
5.5 Alternative Objectives

Besides objective function $OF$ as defined in Section 4.4, specific alternatives are also tested to investigate how they affect the model performance. In all alternatives, the data instances of April, May, June 2011 and June, July, August 2012 were used for testing and the model was run for 3000 seconds per month, while the post processing steps were run for 100000 fails each.

5.5.1 Alternative expression for non delivery and inventory deficit costs

This section examines the model performance when non delivery and inventory deficit costs are modeled in an alternative way. Regarding the non delivery cost, instead of considering the contribution of each product, the objective is expressed as the number of demands whose satisfaction lies within specific percentage intervals. More specifically, AC proposed to use the following intervals:

- $0\% - 80\%$
- $80.1\% - 90\%$

Table 5.8: Model performance for $W_{ndc} : W_{idc} : W_{sc} = 100 : 1 : 0$

<table>
<thead>
<tr>
<th>Data instance</th>
<th>Non delivery costs (million)</th>
<th>Inventory deficit costs (million)</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>April, May, June 2011</td>
<td>44.19</td>
<td>15.52</td>
<td>0.93</td>
<td>60.64</td>
</tr>
<tr>
<td>June, July, August 2012</td>
<td>5</td>
<td>13.91</td>
<td>0.7</td>
<td>19.61</td>
</tr>
</tbody>
</table>
Figure 5.3: Effect of different weights for non delivery and inventory deficit costs on June, July, August 2012

- 90.1% - 95%
- 95.1% - 98%
- 98.1% - 99.9%
- 100%

Similarly, the inventory deficit cost is expressed as the number of products within specific percentage intervals of the corresponding stock target. Again, several percentage intervals are used:

- 0% - 80%
- 80.1% - 90%
- 90.1% - 99.9%
- 100%

So, instead of using $O_{ndc}$ and $O_{idc}$ as defined by Equations 4.15 and 4.16, $O'_{ndc}$ and $O'_{idc}$ are used as defined below:
\[ O'_{ndc} = \sum_{o \in O} ndcModifier_o \cdot (ds_o \leq 0.8 \cdot dem_o) \]
\[ + \sum_{o \in O} ndcModifier_o \cdot (ds_o \leq 0.9 \cdot dem_o) \]
\[ + \sum_{o \in O} ndcModifier_o \cdot (ds_o \leq 0.95 \cdot dem_o) \]
\[ + \sum_{o \in O} ndcModifier_o \cdot (ds_o \leq 0.98 \cdot dem_o) \]
\[ + \sum_{o \in O} ndcModifier_o \cdot (ds_o < dem_o) \] (5.1)

\[ O'_{idc} = \sum_{p \in P, m \in monthRange} (I_{p,m} \leq 0.8 \cdot stockTarget_p) \]
\[ + \sum_{p \in P, m \in monthRange} (I_{p,m} \leq 0.9 \cdot stockTarget_p) \]
\[ + \sum_{p \in P, m \in monthRange} (I_{p,m} < stockTarget_p) \] (5.2)

Note that \( O'_{ndc} \) includes the modifier of each demand \( o \in O \) (ndcModifier\(_o\)), so that the model can take into consideration the strategic importance of each product demand. Also note that in \( O'_{idc} \), there are not so many intervals for higher percentage values, as for the non delivery cost. AC decided to formulate \( O'_{idc} \) that way, since the coverage of stock targets is not as crucial as to satisfy a demand by 100\%. Tables 5.9 and 5.10 present the test results for the data instances of April, May, June 2011 and June, July, August 2012. Each data instance was tested with two different objective weight combinations, to investigate the behavior of the model with these new objectives. In the first case, the weight of non delivery costs \( W'_{ndc} \) and the weight of inventory deficit costs \( W'_{idc} \) get the values 2 and 1 respectively, to check the model behavior when non delivery and inventory deficit costs are of similar importance. Then, in the second case the weights are \( W'_{ndc} = 100 \) and \( W'_{idc} = 1 \). Since Section 5.4 showed that the model behaved better when there was a big difference between these weights, the values of 100 and 1 are used in these tests too. As to \( W_{sc} \), it is reduced even further (to 0.005), because the two alternative objectives \( O'_{ndc} \) and \( O'_{idc} \) get far smaller values than the original. Regarding the model running time, the model was limited to 3000 seconds per month and each post processing step to 100000 fails.

<table>
<thead>
<tr>
<th>Weights</th>
<th>Non delivery costs (million)</th>
<th>Inventory deficit costs (million)</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2:1:0.005</td>
<td>97.4</td>
<td>13.99</td>
<td>1.08</td>
<td>112.47</td>
</tr>
<tr>
<td>100:1:0.005</td>
<td>57.86</td>
<td>11.7</td>
<td>0.96</td>
<td>70.52</td>
</tr>
</tbody>
</table>

Table 5.9: Results of April, May, June 2011 when using alternative objectives

If the results of the two tables are compared to the corresponding results when using the original objective function, it becomes clear that the alternative objectives, as defined by Equations 5.1 and 5.2, do not perform well. This was expected since \( O'_{ndc} \) considers only the satisfaction percentage of each order, and not the unsatisfied amount (expressed
Weights | Non delivery costs (million) | Inventory deficit costs (million) | Setup costs (million) | Total costs (million)
--- | --- | --- | --- | ---
2:1:0.005 | 39.62 | 11.98 | 0.76 | 52.36
100:1:0.005 | 5.22 | 11.24 | 0.69 | 17.15

Table 5.10: Results of June, July, August 2012 when using alternative objectives

in tons) or the contribution $\text{contribution}_p$ of each product $p \in P$. Hence, the model “prefers” to satisfy smaller demands. For instance, consider a demand $A$ of 100 tons and a demand $B$ of 10 tons. According to Equation 5.1, the satisfaction of 81 tons of $A$ and 0 tons of $B$ has the same impact on the objective function, as the satisfaction of 0 tons of $A$ and 8.1 tons of $B$. However, there is a big difference in the resulting non delivery cost of $GO$. As to the contribution, if demands $A$ and $B$ are both 10 tons but $A$ has a bigger contribution, then the satisfaction of $A$ should be more important than the satisfaction of $B$. As said though, $O'_{ndc}$ does not consider $\text{contribution}_p$. Thus, the model will not put more effort on satisfying $A$ instead of $B$, since the impact on the objective function will be the same. Again though, the difference in the non delivery cost of $GO$ will not be negligible. After performing a deeper analysis on the results, it seems that the two cases described are actually the causes for the bad model performance. For instance, in the data instance of April, May, June 2011 there are 16 demands of more than 100 tons each. From these 16 demands, when the weights are $2 : 1 : 0.005$, 8 demands have a satisfaction rate of less than 5%. Also, when the weights are $100 : 1 : 0.005$, there are 6 demands (out of the particular 16), whose satisfaction rate is less than 5%. In contrast, when the original objective function is used and the weights are $100 : 1 : 0.05$ or $2 : 1 : 0.05$, all 16 demands are satisfied by far more than 5%. The same is also observed in the data instance of June, July, August 2012, where there are 8 demands of more than 100 tons. With the alternative objective function, when weights are $2 : 1 : 0.005$ or $100 : 1 : 0.005$, there are 4 and 0 demands (out of the particular 8) respectively with a satisfaction rate of less than 5%. On the other hand, when the original objective function is used and the weights are $100 : 1 : 0.05$ or $2 : 1 : 0.05$, these numbers are 1 and 0 respectively. Hence, it becomes obvious that the alternative objectives $O'_{ndc}$ and $O'_{idc}$ do not capture accurately the importance of the various demands, leading to far worse results than the original objective function.

5.5.2 Alternative expression for setup costs

In this subsection $O_{sc}$ in $OF$ is replaced by $O'_{sc}$ as presented in Section 4.4, with a weight $W'_{sc} = 1$. Regarding the weights of $O_{ndc}$ and $O_{idc}$, they are set to 100 and 1 respectively. As explained, in $O'_{sc}$ each changeover is charged with a cost of $\text{coCost}$. Besides this, AC wished to investigate the effect of $O'_{sc}$ to the solution quality in cases where changeovers cost $0.5 \times \text{coCost}$ or $2 \times \text{coCost}$ euros each, irrespective of their type. Tables 5.11 and 5.12 present the results for April, May, June 2011 and June, July, August 2012 respectively. Also, for each data instance the results when using $OF$ are presented again for comparison.

<table>
<thead>
<tr>
<th>Cost per changeover</th>
<th>Non del. &amp; inv. deficit costs (million)</th>
<th>Number of changeovers</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
<th>Setup costs Total costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.5 \times \text{coCost}$</td>
<td>65.94</td>
<td>100</td>
<td>0.5</td>
<td>66.44</td>
<td>0.75%</td>
</tr>
<tr>
<td>$\text{coCost}$</td>
<td>63.64</td>
<td>82</td>
<td>0.82</td>
<td>64.46</td>
<td>1.27%</td>
</tr>
<tr>
<td>$2 \times \text{coCost}$</td>
<td>64.55</td>
<td>85</td>
<td>1.7</td>
<td>66.25</td>
<td>2.57%</td>
</tr>
<tr>
<td>Original $OF$</td>
<td>47.94</td>
<td>77</td>
<td>0.77</td>
<td>48.71</td>
<td>1.58%</td>
</tr>
</tbody>
</table>

Table 5.11: Tests with different cost per changeover on April, May, June 2011
<table>
<thead>
<tr>
<th>Cost per changeover</th>
<th>Non del. &amp; inv. deficit costs (million)</th>
<th>Number of changeovers</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
<th>Setup costs Total costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5 × coCost</td>
<td>14.07</td>
<td>72</td>
<td>0.36</td>
<td>14.43</td>
<td>2.49%</td>
</tr>
<tr>
<td>coCost</td>
<td>14.21</td>
<td>72</td>
<td>0.72</td>
<td>14.93</td>
<td>4.82%</td>
</tr>
<tr>
<td>2 × coCost</td>
<td>13.45</td>
<td>59</td>
<td>1.18</td>
<td>14.63</td>
<td>8.06%</td>
</tr>
<tr>
<td>Original OF</td>
<td>12.19</td>
<td>70</td>
<td>0.7</td>
<td>12.89</td>
<td>5.43%</td>
</tr>
</tbody>
</table>

Table 5.12: Tests with different cost per changeover on June, July, August 2012

The results indicate that the increase in the cost of changeovers does not have a very big effect in the sum of non delivery and inventory deficit costs. Furthermore, there is no standard pattern in how the summation of these costs changes, while increasing the cost of setups. For instance, the highest sum of non delivery and inventory deficit costs for April, May, June 2011 is when the changeovers cost 0.5 × coCost euros each, while for June, July, August 2012 the highest sum is for coCost euros per changeover.

Regarding the setup cost, it obviously increases as the cost per changeover increases as well. For both instances this seems to have a positive effect or no effect at all on the number of changeovers, with an exception of the data instance of April, May, June 2011, where the number of changeovers increases when the cost per changeover is increased from coCost euros to 2 × coCost euros each. However, the Setup costs Total costs ratio is increasing as the cost per changeover increases.

As to the setup time, there is no pattern for this either. In the data instance of 2011, the total duration for costs of 0.5 × coCost, coCost, and 2 × coCost euros is 1023, 734, and 802 hours respectively, while in the data instance of 2012 the same numbers are 662, 827, and 485 hours respectively. The existence of no relation between the setup durations of the three model runs of each data instance was expected, since O′sc only considers the number of changeovers and not their duration. Finally, a last remark is that all these tests do not manage to give a solution of similar quality to the solution quality obtained when using the original OF.

5.6 Effect of Random Seed

As explained in Section 4.5.2, in OPL the programmer is able to direct the search of the algorithm. Besides search phases, parameters can be set to choose between different search strategies (e.g.: Depth-first search, Restart search, or Multi-point search) or to change the value of random seed. For some strategies, the search uses randomization and by changing the seed of the random generator, the programmer can tune the optimizer. The effect of random seed to the solution quality was investigated and it is presented in this section. In total, five different values of random seed were tested in the data instances of April, May, June 2011 and June, July, August 2012. In all tests the running time of the model was limited to 3000 seconds per month and each post processing step was run until 100000 fails. As to the objective weights, Wnde = 100, Wide = 1, and Wsc = 0.05. The results of the two data instances are presented in Tables 5.13 and 5.14.

From the results of April, May, June 2011 it seems that the increase of random seed has a negative effect on the solution quality. However, this is not the case for the data instance of June, July, August 2012, where a seed equal to 100 or 500 gives better results than the default value of random seed. For random seed equal to 200 or 1000 though, the results are worse. Since there is no improvement in both data instances with the same seed values, it means that the effect of random seed on the model performance is strongly related to the data instance. So, there are no safe results that can be extracted by these tests. However, the idea tested below is the following.
Since the results of June, July, August 2012 for seeds equal to 100 and 500 are better, it means that the model using the new seed values, gets to a better solution faster than the model with the default seed. Thus, instead of running the model for 3000 seconds per month using the default seed, we can run it two times, once with seed equal to 100 and another with seed equal to 500, with a time limit of 1500 seconds per month. Then we can just keep the best result out of the two. If indeed the model gets sooner to better results, then it is probable that one (or both) of the results with different seed will give better result than the one obtained with default seed value. Note that the total running time of the two model runs with the new seed values, is almost equal to the running time of the model with the default seed. The difference lies on the fact that in the case of different seeds, since the model is run twice, the post processing steps are also run twice. Thus almost the same amount of time is spent, but instead of obtaining a single result, two are obtained. Table 5.15 presents the results after running the model for 1500 seconds per month with seed values of 100 and 500. Note that only the data instance of June, July, August 2012 was tested, since for April, May, June 2011 all tests with different random seed values gave worse results than the default seed.

<table>
<thead>
<tr>
<th>Seed</th>
<th>Non delivery costs (million)</th>
<th>Inventory deficit costs (million)</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>4.74</td>
<td>10.5</td>
<td>0.64</td>
<td>15.9</td>
</tr>
<tr>
<td>500</td>
<td>6.5</td>
<td>13</td>
<td>0.69</td>
<td>20.3</td>
</tr>
</tbody>
</table>

Table 5.15: Tests on June, July, August 2012 with time limit of 1500 seconds per month

From the results of Table 5.15 it is clear that the performance improvement achieved by using different seed values when running the data instance of June, July, August 2012 for 3000 seconds, is not enough to give better results on half the running time. Thus the idea of running the model twice on half the running time with different seed values, does not provide better results than running the model for the whole run time using the default seed value. Furthermore, since a particular seed value does not improve performance on all data instances, we cannot conclude on which random seed is best for the model.
5.7 Constraint Relaxation

AC proposed several constraints to be relaxed or totally removed, to investigate how these affect the quality of the solution. By doing so, AC can find out whether some particular modifications in the production lines or the working shifts of the staff can have a positive impact. In order to have more concrete results, each case is tested in three different single-month data instances, namely April 2012, June 2012, and August 2012. For all runs and data instances, all model parameters (except from the relaxed/disabled constraints) are the same, in order to make the comparison between different cases possible. Furthermore, the fixed tasks are not considered in these cases, in order to have a better view of the effect of each relaxed or removed constraint. As to the running limit of the model, each data instance was run for 750000 fails. Finally note that each constraint relaxation or removal was applied separately and that Equations 4.28 and 4.29 as well as the second and third step of the post processing method, were not included in the model.

Die changeovers: According to non-technical Constraint 2, die changeovers are not allowed during the weekends. This constraint is very limiting and has a big impact in the quality of the produced solution. Thus, it was decided to test the model without this constraint.

Starting production during weekdays: Finally, another limiting constraint is non-technical Constraint 1, which states that the production of some specific products can only start during the weekdays. Since this constraint seems to limit the quality of the model considerably, it was decided to be removed and investigate its impact.

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Data instance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>April 2012</td>
</tr>
<tr>
<td></td>
<td>total costs (million)</td>
</tr>
<tr>
<td>Original</td>
<td>10.1</td>
</tr>
<tr>
<td>Die Changeovers</td>
<td>6.96 (-31%)</td>
</tr>
<tr>
<td>Starting production during weekdays</td>
<td>9.19 (-9%)</td>
</tr>
</tbody>
</table>

Table 5.16: Results after relaxing/removing constraints

The results of the aforementioned constraint relaxations/removals are presented in Table 5.16 and Figure 5.4.

As to the constraint for the die changeovers, its removal has a considerable impact in the model performance leading to a mean cost reduction of 29%. This result was not a surprise as the particular constraint concerns all products (since all products can have a die changeover before them).

Finally, removing the non-technical Constraint 1 led to an improvement of the model solution in all data instances. Similarly to above, the rate of improvement is not the same for all data instances, but it depends on the special attributes of each one.

Please note that several other technical constraints were relaxed to test the model’s behavior and also, an extensive analysis is performed. However, since these are considered classified by AC, they are not mentioned in this version of the report.
5.8 Test on Larger Data Instance

Besides the three-months data instances used so far, the CP model is also tested on a larger data instance. That way the stock development of products is investigated, and whether the model is capable of maintaining low inventory deficit costs for a long time period. The conclusions from this test are important regarding the usefulness of the model for AC. The only 6-months data instance available is April - September 2012. Table 5.17 presents the results after running the model for 3000 seconds per month and each post processing step for 100000 fails. As to the objective weights of OF the following were used, since they give the best results according to Section 5.4: $W_{ndc} : W_{Idc} : W_{sc} = 100 : 0.1 : 0.05$.

<table>
<thead>
<tr>
<th>Month</th>
<th>Non delivery costs (million)</th>
<th>Inventory deficit costs (million)</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>April</td>
<td>1.61</td>
<td>4</td>
<td>0.25</td>
<td>5.86</td>
</tr>
<tr>
<td>May</td>
<td>1.54</td>
<td>4.07</td>
<td>0.22</td>
<td>5.83</td>
</tr>
<tr>
<td>June</td>
<td>0</td>
<td>4.04</td>
<td>0.23</td>
<td>4.27</td>
</tr>
<tr>
<td>July</td>
<td>0.75</td>
<td>5.74</td>
<td>0.19</td>
<td>6.68</td>
</tr>
<tr>
<td>August</td>
<td>1.37</td>
<td>5.85</td>
<td>0.19</td>
<td>7.41</td>
</tr>
<tr>
<td>September</td>
<td>0.47</td>
<td>5.79</td>
<td>0.22</td>
<td>6.48</td>
</tr>
<tr>
<td>Total</td>
<td>5.74</td>
<td>29.49</td>
<td>1.3</td>
<td>36.53</td>
</tr>
</tbody>
</table>

Table 5.17: Results of the 6-month data instance

As seen by the results, initially the model manages to maintain inventory deficit costs at about 4 million euro. More specifically, from April to May, inventory deficit costs are increased just by 70000 euro, but then they are decreased to 4.04 million by the end of June. Unfortunately though, the inventory deficit cost is increased by 42% approximately, to 5.74 million, at the end of July. However, a big increase (approximately 52%) between the same months is also noticed in the instance of June, July, August 2012, when run for the same time limit and the same weights (Table 5.18). Hence, this particular increase may not be due to bad model performance, but because of the particular data instance. The difference is that in the case of Table 5.18, the model manages to reduce inventory
deficit cost in the next month, while in the 6-months instance, this is not managed. Back to Table 5.17, during the months after July, there are only small fluctuations in the inventory deficit cost. Hence, it seems that the model can generally maintain the inventory deficit cost level, since it is not increased in every month. In fact there are two months, where a small decrease is also noticed. However, the big increase in July is an indication that the model may not be able to handle it so well. Unfortunately, a bigger instance was not available to investigate the development of inventory deficit cost and whether the model would for example manage to lower it back to 4 million again.

<table>
<thead>
<tr>
<th>Month</th>
<th>Non delivery costs (million)</th>
<th>Inventory deficit costs (million)</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>June</td>
<td>0</td>
<td>2.54</td>
<td>0.26</td>
<td>2.8</td>
</tr>
<tr>
<td>July</td>
<td>0.19</td>
<td>3.86</td>
<td>0.24</td>
<td>4.29</td>
</tr>
<tr>
<td>August</td>
<td>0</td>
<td>3</td>
<td>0.23</td>
<td>3.23</td>
</tr>
<tr>
<td>Total</td>
<td>0.19</td>
<td>9.4</td>
<td>0.73</td>
<td>10.3</td>
</tr>
</tbody>
</table>

Table 5.18: Results of June, July, August 2012 with weights $W_{ndc} : W_{idc} : W_{sc} = 100 : 0.1 : 0.05$

Note that a direct comparison of the results of months June, July and August between the two data instances is not possible. This because the forecast data for months April and May used in April - September 2012, are not the same with the actual demands of these months, that led to the inventory levels that are used as starting point in the data instance of June, July, August 2012. This is owed to the fact that the forecasts of a particular month are changing continuously and they are only an approximation of the actual demand. Hence the difference lies to the fact that the inventory levels used in the data instance of June, July, August 2012 are the actual levels at the beginning of June, as they were determined after satisfying the real demands for month May, while the inventory levels used in April - September 2012 for month June are the result of the schedule that was created while considering the forecasts for April and May, and not the actual demands.

5.9 Comparison of Model and Manual Solution

The comparison of a model solution to the manual planning created for the same data instance, is very important to the evaluation of the model performance. Hence, the manually created planning for the time period of June, July, August 2012, is compared to the model solution. However, for the model solution to be comparable to the manual, a few more fixed tasks needed to be added to the model. Thus, the solutions presented above could not be used for the comparison, and a new run was made. In Section 5.4 it is shown that the best results are obtained when the objective weights are $W_{ndc} : W_{idc} : W_{sc} = 100 : 0.1 : 0.05$. Also Section 5.3 shows that when the time limit for the sub model is increased from 3000 seconds to 4000 seconds per month, the solution quality for June, July, August 2012 is improved by approximately 26%. Hence, after adding the additional fixed tasks, the model is run with weights $W_{ndc} : W_{idc} : W_{sc} = 100 : 0.1 : 0.05$, for 4000 seconds per month and each post processing step for 100000 fails. Tables 5.19 and 5.20 give the cost results of the two solutions and then Table 5.21 gives some additional data about the two schedules.

As it is also explained in Section 5.8, the data used by the model are only forecasts and not the actual demands during the months of the particular data instance. Since these forecasts change continuously, some of the decisions taken in the manual planning, may correspond to slightly different demand amounts than the forecasts used by the model. Hence, in reality, the performance improvement achieved by the model may not be as
big as it seems in Tables 5.19 and 5.20. However, this comparison constitutes at least an indication that the performance of the proposed model is quite satisfying, and that its quality can be compared to the quality of a manual schedule.

<table>
<thead>
<tr>
<th>Month</th>
<th>Non delivery costs (million)</th>
<th>Inventory deficit costs (million)</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>June</td>
<td>0</td>
<td>1.8</td>
<td>0.24</td>
<td>2.04</td>
</tr>
<tr>
<td>July</td>
<td>0</td>
<td>2.45</td>
<td>0.23</td>
<td>2.68</td>
</tr>
<tr>
<td>August</td>
<td>0</td>
<td>2.7</td>
<td>0.23</td>
<td>2.93</td>
</tr>
<tr>
<td>Total</td>
<td>0</td>
<td>6.95</td>
<td>0.7</td>
<td>7.65</td>
</tr>
</tbody>
</table>

Table 5.19: Model solution

<table>
<thead>
<tr>
<th>Month</th>
<th>Non delivery costs (million)</th>
<th>Inventory deficit costs (million)</th>
<th>Setup costs (million)</th>
<th>Total costs (million)</th>
</tr>
</thead>
<tbody>
<tr>
<td>June</td>
<td>1.01</td>
<td>2.76</td>
<td>0.2</td>
<td>3.97</td>
</tr>
<tr>
<td>July</td>
<td>0.8</td>
<td>3.23</td>
<td>0.14</td>
<td>4.17</td>
</tr>
<tr>
<td>August</td>
<td>0.68</td>
<td>3.23</td>
<td>0.3</td>
<td>4.21</td>
</tr>
<tr>
<td>Total</td>
<td>2.49</td>
<td>9.22</td>
<td>0.64</td>
<td>12.35</td>
</tr>
</tbody>
</table>

Table 5.20: Manual solution

<table>
<thead>
<tr>
<th></th>
<th>Model solution</th>
<th>Manual solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Production (tons)</td>
<td>4111.9</td>
<td>4278.3</td>
</tr>
<tr>
<td>Total delivered amount (tons)</td>
<td>3746</td>
<td>3647.55</td>
</tr>
<tr>
<td>Die changeovers</td>
<td>23</td>
<td>3</td>
</tr>
<tr>
<td>Total changeovers (including dies)</td>
<td>70</td>
<td>64</td>
</tr>
</tbody>
</table>

Table 5.21: Additional data for model and manual solutions

As shown, the model performs better in terms of non delivery cost, inventory deficit cost, as well as the total costs. More specifically, the model solution is better than the manual by approximately 38%. Especially in terms of non delivery cost, the model performs very well, since it manages to deliver all demands, while in the manual planning some demands are not fully satisfied.

Table 5.21 shows that the total amount produced by the model solution is a little less than the amount produced by the manual planning, and still, more tons are delivered and the stock targets of more products are covered. This means that the model took the correct choices on which products to produce and on which amounts. However the difference is not that big, and it is probable that it is owed to the use of outdated forecast data.

A big drawback of the model solution is that it schedules more changeovers than the manual solution. Especially for the time consuming die changeovers, the model schedules 20 more than the manual planning. Although it was not possible to calculate the exact total duration of changeovers in the manual planning, it is certain that it is less than the changeover duration proposed by the model solution. The increased number of changeovers and the increased duration, comes at a cost of less available production time and higher setup costs (0.7 instead of 0.64 million). However, this difference is not enough to outweigh the cost reduction achieved in non delivery and inventory deficit costs. Also, the model’s
choice for more changeovers is supported by the decisions taken by the model proposed in [29]. In [29], the die changes and the total setup duration were again higher than in the corresponding manual planning. Since both models show a performance improvement despite the high number of changeovers, it indicates that the increase of changeovers may lead to better results because of a more efficient use of the resources in the production facility of AC.

Finally an issue of the model is that it cannot handle any small gaps that are left empty between fixed tasks. In the manual planning, the constraint regarding \( mpt \) may be violated in some cases, to allow production for shorter time periods. Thus, any small gaps between consecutive fixed tasks are easily filled. However, this is not allowed in the model, resulting in small gaps in the solution. The same happens when the free time between the end of a fixed task and the end of the planning horizon is less than \( mpt \). Even if there is an additional month in the data instance, the model will not be able to schedule any production, because it does not consider the next month. This is obviously another drawback of the decomposition method.
Chapter 6

Final Improvements

This chapter presents some last minute modifications that were tested to investigate whether they can improve the performance of the CP model. These are separated from Section 4.5, since they were performed after the tests presented in Chapter 5. Hence, any indications of performance improvement obtained by these tests, are left as future work. Each modification was tested on the data instances of April, May, June 2011 and June, July, August 2012, where the model ran for 3000 seconds and each post processing step for 100000 fails. As to the objective weights, they were $W_{ndc} : W_{idc} : W_{sc} = 100 : 1 : 0.05$. Except from the parameter or constraint that was tested, everything else was as described in Section 4.7.

Modeling of changeover tasks: In Section 4.3, the optional intervals $cotType_{i,m,t}$, where $i \in I^{spin}$, $m \in monthRange$, $t \in \{0, 3, 7, 24\}$ are defined, and then they are used in Equations 4.5 and 4.6. The idea for the potential improvement here, is that instead of defining 4 $cotType_{i,m,t}$ intervals for each changeover task, only 2 can be defined, where the first can be of size ranging from 0 to 7, while the second can only have a size of 24. That way, for each present spinning task interval $spt_{i,m}$, $i \in I^{spin}$, $m \in monthRange$, 2 less $cotType_{i,m,t}$ intervals are created. Thus the model has less variables in total. However, this approach has the drawback that now the size of the first type of $cotType_{i,m,t}$ variables has a variable size ranging between $[0, 7]$, adding complexity to the model. As to the constraints described by Equations 4.5 and 4.6, no change is needed.

MIP/CP decomposition: Although it is shown in Section 4.5.1 that the MIP/CP decomposition deteriorated performance, some further investigation is made. The two models are again the same as in Section 4.5.1, and the difference only lies on how the CP model uses the solution of the MIP model. As a first test, instead of using the MIP solution just as a starting point, the inventory levels and the demand satisfaction amounts decided by the MIP model are set as minimum values to the corresponding decision variables of the CP model. As an alternative to the above, the inventory levels and the demand satisfaction amounts decided by the MIP model are again used as minimum values to the corresponding decision variables of the CP model, but now the MIP solution is not used as a starting point too.

Table 6.1 presents the results for the modification regarding the changeovers, and it is made clear that the initial way of modeling performs better. Hence it is preferable to have more interval variables (approximately 1500 more for April, May, June 2011) with predetermined size, instead of having less with variable size.

As to the ideas for the MIP/CP decomposition, the tests showed that when the inventory levels and the demand satisfaction amounts decided by the MIP model, are used as minimum values to the corresponding variables of the CP model, then no solution can
be found. More specifically, the CP model found no solution even after running for 3000 seconds. Hence, further tests were performed, where the minimum values of the CP variables were only a fraction of the values decided by the MIP model. Testing, where the MIP solution was also set as a starting point to the CP model, showed that the CP model found a solution for the data instance of April, May, June 2011, only when the minimum values of the CP variables were set to 20% of the values decided by the MIP model. As to the data instance of June, July, August 2012, the CP model found a solution when the minimum values were equal to the 30% of the corresponding MIP values. In the case where no starting point was set to the CP model, for the data instance of April, May, June 2011 no solution was found even at 20%, while for June, July, August 2012, a solution was found at 30%, but it was worse than when a starting point was set. Note though, that when it is said that the CP model finds no solution, it means that it cannot find a solution for all three months of the data instance. For example, in the data instance of June, July, August 2012, a solution was found for the month of June when the minimum values were set to 40%, but then the CP model failed to find a solution for the month of July. Also note that only fractions that are multiples of 10 were tested (i.e., 100%, 90%, 80%, etc.). The results are presented in Table 6.2 along with the results of the model as described in Section 4.7, for comparison. As shown, the MIP/CP model is outperformed by the CP model.

<table>
<thead>
<tr>
<th></th>
<th>Data instance</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>April, May, June 2011</td>
<td>June, July, Aug. 2012</td>
</tr>
<tr>
<td></td>
<td>total costs (million)</td>
<td>total costs (million)</td>
</tr>
<tr>
<td>Original</td>
<td>48.71</td>
<td>12.9</td>
</tr>
<tr>
<td>Changeover tasks</td>
<td>59.02 (+21.1%)</td>
<td>16.89 (+30.9%)</td>
</tr>
</tbody>
</table>

Table 6.1: Test results of the modifications changeover tasks.

<table>
<thead>
<tr>
<th></th>
<th>Data instance</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>April, May, June 2011</td>
<td>June, July, Aug. 2012</td>
</tr>
<tr>
<td></td>
<td>total costs (million)</td>
<td>total costs (million)</td>
</tr>
<tr>
<td>Original</td>
<td>48.71</td>
<td>12.9</td>
</tr>
<tr>
<td>With Starting point</td>
<td>71.44 (+46.7%)</td>
<td>14.98 (+16.1%)</td>
</tr>
<tr>
<td>No starting point</td>
<td>No solution available</td>
<td>17.65 (+36.8%)</td>
</tr>
</tbody>
</table>

Table 6.2: Test results using MIP/CP decomposition

The fact that the values found by the MIP model have to be set to 30% or even 20% for the CP model to find a solution, indicates that the problem approximation that is made in the MIP model is not precise enough. In other words, the MIP model disregards many constraints, resulting in a solution that is far from a realistic one. Hence, in order to have an efficient MIP/CP decomposition model, further work has to be done towards a more accurate MIP model.
Chapter 7

Conclusion

The main goal of this project was to develop a model for solving the production planning and scheduling problem at AC. The initially proposed solution by [29] proved to not be usable in practice by AC, since it was non-deterministic and gave results of high variability. Thus we aimed at the development of well-performing deterministic models that would have practical value for AC. These goals were accomplished by developing two totally new deterministic models, one based on Mixed Integer Programming (MIP) and the other on Constraint Programming (CP), using the previous solution as a starting point. An additional project goal was the comparison of the two approaches used in the models.

The MIP model we developed, was based on the mathematical formulation presented in [29], and proved to be unable to handle the complexity of the problem found in AC, as it could not run for the full data instances. In order to investigate whether the model works correctly, as well as get an idea of the complexity that it can handle, smaller data instances were also tested. As explained in Section 3.6, the MIP model gives correct solutions (i.e., all constraints are satisfied) and it also succeeds in identifying the best decisions for each scenario. However, even for the smallest data instances the model takes too long to prove optimality, indicating poor model performance. This becomes more severe in larger data instances, and at the full data instances the model does not even manage to start searching for a solution. In an attempt to improve performance, linear forms of non linear constraints were created, but still no satisfying improvement was achieved.

As to the CP model, although its initial performance was not good either, it is shown to perform well after applying several modifications. These modifications include the decomposition of the main model, the use of search phases, as well as the limitation of the search space. An unexpected improvement was achieved by using meta-constraints instead of regular ones, leading to an improvement of almost 9%. Furthermore, we proposed a post processing method that boosts the model performance even further. This method comprises of three steps, whose main objectives are to fill any gaps in the schedule, as well as reduce changeovers. Tests showed an average cost reduction of 14%, as well as the removal of any unnecessary changeovers.

In order to investigate whether the CP model proposes the optimal decisions in different what-if scenarios, the six validation cases as described in [29] were tested. The results indicate that the CP model not only takes the correct decisions, but it also performs better than the model proposed in [29]. However a direct comparison of the two models was not possible, since several constraints of AC have changed since [29]. To evaluate the model behavior and performance, several other tests were executed on different running times and objective weights, alternative objective functions, different algorithm seeds, as well as the relaxation of particular constraints. All these tests proved to be fruitful, as they gave important information about the quality of the model. For instance, as shown in Section 5.3, it is probable that an increase in running time will result in worse overall
solution quality for a given 3-months data instance. This is owed to the fact that when
the model optimizes a particular month, it does not consider the following months of the
data instance. Thus, for a particular month, the model may give a solution that although
it is quite good for the current month, the decisions taken have a very negative impact
to the future months. A possible solution would be the model to take into account the
product demands of the following months. By doing so, it may avoid taking decisions that
are quite detrimental for the future months. Another issue of the model is that it cannot
handle efficiently cases where non delivery costs and inventory deficit costs are of similar
importance for the objective function. As shown in Section 5.4, the model performance
is best when the weight factor of non delivery costs is 1000 times bigger than the weight
factor of inventory deficit costs.

Regarding the tests with relaxed constraints, they were all proposed by AC and their
purpose was to investigate how they affect the solution quality, and whether particular
decisions regarding the organization of AC can have a positive impact. The results are
quite interesting, as they suggest that AC should hire for example more staff to enable the
delete changeovers during the weekends. The tests indicate that by doing so, AC can achieve a
cost reduction by up to 29%.

To evaluate the practical value of the CP model, a test on a longer data instance was
also performed. That way, it was investigated whether the model is able to maintain
stocks at a satisfying level. Despite an increase during a particular month of the 6-months
data instance, the results showed that in general the model satisfies the demands without
depleting inventory levels. Furthermore, the model was evaluated by comparing its outcome
to the corresponding manual schedule. The comparison showed an improvement in the
total costs by approximately 38%. However, a part of this improvement is only due
to the use of outdated forecast data by the model. This comparison also shows that the
model solution contains many changeovers, despite the considerable reduction achieved by
the post processing method. This outcome though, confirms the model of [29], whose results contain again a large number of changeovers. Hence it is probable that the increase of changeovers leads to better results indeed, since it allows a more efficient use of the
resources in the production facility of AC.

Some last minute changes were also tested to check whether they can improve the
model performance. The initial model proved to give better results however. These tests
also showed that the MIP model proposed in Section 4.5.1 for the MIP/CP decomposition
model is not accurate enough, and that more constraints should be included in it. Unfor-
tunately, due to time limitations, no further investigation was made on this.

As to the project goals, the final model is able to give deterministic results, provided
that the search of the algorithm is limited by the number of fails. Regarding the compar-
ison of the two approaches, the MIP model is obviously worse, since it cannot handle the
problem of AC at its full complexity. Another disadvantage of the MIP approach is that
in general, it was more difficult to model the constraints of AC, as well as verifying their
correctness. Finally, the comparison of the model solution to a manual schedule, as well
as the other tests and the careful analysis of the detailed schedules, prove that the model
is producing solutions that can have practical value for AC.

7.1 Future Work

As shown by the thorough testing that was performed, there is still room for improving
the model. Furthermore, the issue with the objective weights has not been tackled suc-
cessfully yet, and there are still several tests that need to be performed regarding model
improvements or experiments that would be interesting for AC. Thus, the following could
be performed as future work:
• Investigate more thoroughly whether the model suggestion for more changeovers (as done in Section 5.9) is optimal.

• Improve the model to be able to handle cases where non delivery and inventory deficit costs are of similar importance in the objective function.

• Apply the appropriate modifications to make the decomposition model capable of considering the following months of the planning horizon. For instance, the model can consider any big demands of the next months (for example, demands above 100 tons) and try to schedule production for the corresponding products, if there is availability.

• According to literature, the decomposition of a problem to a MIP and a CP model is very promising. However, as shown in Chapter 6, the MIP model of the MIP/CP model is not accurate enough. Thus, more constraints should be included in it.

• Test alternative ways of modeling the constraints of the CP model, to investigate whether performance can be improved any further.

• Continue enriching the model with more constraints and optimization criteria (like transportation costs) to make it more suitable of capturing the problem of AC at its full extent.
Bibliography


