Optimal Scheduling of Refinery Crude-Oil Operations

A DISSERTATION

Submitted to the Graduate School

in Partial Fulfillment of the Requirements

for the degree of

Doctor of Philosophy

in

Chemical Engineering

by

Sylvain Mouret

Carnegie Mellon University

Pittsburgh, Pennsylvania

December, 2010

Acknowledgments

First of all, I would like to express my most sincere gratitude to my advisor Professor Ignacio E. Grossmann for his inestimable guidance and support over the course of my Ph.D. He has managed to create a productive yet friendly environment and proved to be an abundant source of knowledge for myself. I cannot thank him enough for his confidence in me and his deep implication in my studies and in my life.

Besides my advisor, I would like to thank my thesis committee members – Professors Lorenz Biegler, Nikolaos Sahinidis, John Hooker, and Willem-Jan van Hoeve for their time and valuable comments.

I would like to thank Pierre Pestiaux, my supervisor at Total, whose strong commitment to the project and never-ending enthusiasm has made this thesis possible.

I would also like to thank Philippe Bonnelle for bringing his experience and his insightful suggestions into the project as well as other collaborators at SOG and CReG, for their useful feedback on my work and friendly support. Furthermore, I am grateful to Total Refining & Marketing for financial support of this project.

I wish to express my thankfulness for all my past and present workmates in the PSE group for setting a productive mood in the office and a diverting atmosphere out of work. Among them I would like to specifically mention Rosanna Franco, Gonzalo Guillén Gosálbez, Ricardo Lima, Rodrigo López-Negrete de la Fuente, Mariano Martin, Roger Rocha, Sebastian Terrazas, and Victor Zavala with whom I share many unforgettable memories.

I would also like to thank my fellow football and tennis teammates, Tarot card players, French speaking lunchers, barbecue grillers, etc... who made my Pittsburgh experience a very enjoyable one.

I want to express my gratitude to my family who has always been there when I needed them, and to my 18-month-old niece Anna for being so cute and joyful.

Last but not least, I cannot thank enough my beloved fiancée Charlotte for her patience and for standing by me during the past three and a half years. Her unconditional love is never to be forgotten.

Abstract

This thesis deals with the development of mathematical models and algorithms for optimizing refinery crude-oil operations schedules. The problem can be posed as a mixed-integer nonlinear program (MINLP), thus combining two major challenges of operations research: combinatorial search and global optimization.

First, we propose a unified modeling approach for scheduling problems that aims at bridging the gaps between four different time representations using the general concept of priority-slots. For each time representation, an MILP formulation is derived and strengthened using the maximal cliques and bicliques of the non-overlapping graph. Additionally, we present three solution methods to obtain global optimal or near-optimal solutions. The scheduling approach is applied to single-stage and multi-stage batch scheduling problems as well as a crude-oil operations scheduling problem maximizing the gross margin of the distilled crude-oils.

In order to solve the crude-oil scheduling MINLP, we introduce a two-step MILP-NLP procedure. The solution approach benefits from a very tight upper bound provided by the first stage MILP while the second stage NLP is used to obtain a feasible solution.

Next, we detail the application of the single-operation sequencing time representation to the crude-oil operations scheduling problem. As this time representation displays many symmetric solutions, we introduce a symmetry-breaking sequencing rule expressed as a deterministic finite automaton in order to efficiently restrict the set of feasible solutions.

Furthermore, we propose to integrate constraint programming (CP) techniques to the branch & cut search to dynamically improve the linear relaxation of a crude-oil operations scheduling problem minimizing the total logistics costs expressed as a bilinear objective. CP is used to derived tight McCormick convex envelopes for each node subproblem thus reducing the optimality gap for the MINLP.

Finally, the refinery planning and crude-oil scheduling problems are simultaneously solved using a Lagrangian decomposition procedure based on dualizing the constraint linking crude distillation feedstocks in each subproblem. A new hybrid dual problem is proposed to update the Lagrange multipliers, while a simple heuristic strategy is presented in order to obtain feasible solutions to the full-space MINLP. The approach is successfully applied to a small case study and a larger refinery problem.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Optimization in the oil refining industry began with the use of linear programming (LP) to perform process and economic analysis of industrial plants (see Garvin et al., 1957; Manne, 1958). Many refinery problems are now addressed with algorithms based on mathematical models: refinery planning, crude-oil operations scheduling, final product blending, crude-oil transportation, final product shipping, and profitability improvement plans (for instance, see Pinto et al., 2000). In general, problem-specific techniques are used to solve each model independently from the others. The goal of this thesis is to develop a general methodology towards enterprise-wide optimization of oil refineries (Grossmann, 2005). Due to the structural diversity of the problems to be solved, the challenge is to effectively integrate different optimization techniques in order to generate near-optimal enterprise-wide solutions. The industrial applications addressed in this thesis are related to single-stage and multi-stage batch processes, medium-term planning of refining operations and short-term scheduling of crude-oil operations. The objectives of the thesis are as follows:

1. Develop a unified modeling approach for solving process scheduling problems
2. Apply the proposed time representations to schedule and optimize batch processes and crude-oil operations
3. Develop and implement general solution methods to effectively solve such problems and obtain near-optimal solutions with rigorous optimality estimates
4. Develop a method for integrating mixed-integer linear programming and constraint programming for improving the linear relaxation of mixed-integer nonlinear programs
5. Apply advanced Lagrangian decomposition techniques to simultaneously solve refinery planning and crude-oil operations scheduling problems

In this chapter, an overview of single-stage and multi-stage batch scheduling problems is presented followed by a description of the refinery planning and crude-oil scheduling problems. The different optimization techniques used are then reviewed and we conclude with an overview of the chapters in the thesis.

1.1 Single-Stage and Multi-Stage Batch Scheduling

The chemical industry has been marked by an increase of product diversification, which in turn has led to an increase in the complexity of operations of plant facilities. Chemical companies are now facing the challenge of meeting global demands of multiple products while increasing plant capacities to achieve economies of scale (Wassick, 2009). Operating optimally such plants can be non-trivial as decision-makers have to account for demand deadlines, process constraints, and limited resources. Therefore, the scheduling of chemical processes has received much attention over the past 20 years. Two major categories of processes have been outlined and addressed: sequential and network-based processes (see process classification in Méndez et al., 2006a). The main difference lies in the fact that network processes may display recycle loops which sequential processes do not.

Recently, several research groups have reviewed the different trends in process scheduling for general purpose plants. Floudas and Lin (2004) provided an extensive comparison of discrete-time and continuous-time formulations. In Méndez et al. (2006a), a complete classification of scheduling approaches is presented in addition to the process classification. In this thesis, we study single-stage and multi-stage batch scheduling problems. Figure 1.1 depicts a typical multi-stage batch process. A finite number of batches with fixed sizes have to be processed going through a given set of successive stages. In each stage, a finite set of parallel units is available. The processing times for each batch and each stage may be unit-dependent. Stages with limited resources or high processing times are often called *bottleneck* stages. Different policies can be used for interstage storage: unlimited intermediate storage (UIS), finite intermediate storage (FIS), no intermediate storage (NIS), and zero-wait (ZW).

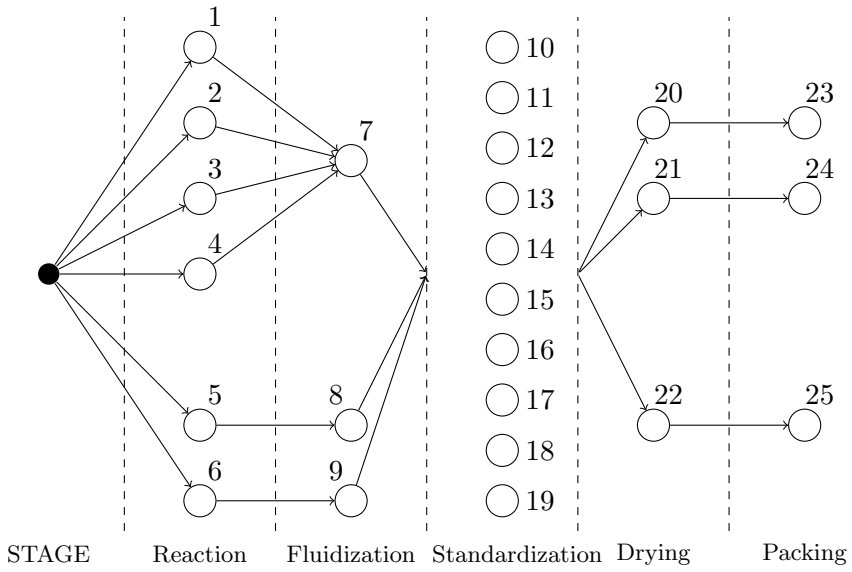


Figure 1.1: A typical multi-stage batch process from Pinto and Grossmann (1995).

Two main scheduling approaches have been developed to solve batch scheduling problems: precedence-based and slot-based models.

The idea of using disjunctive programming principles (see Balas, 1985) to solve chemical process scheduling problems was initially introduced by Cerdá et al. (1997) who proposed a mathematical model for scheduling single-stage multiproduct batch plants. Among others, Gupta and Karimi (2003); Méndez and Cerdá (2007); Marchetti and Cerdá (2009) have successively improved and extended precedence-based formulations and applied them to multistage batch scheduling problems. They consider complex scheduling features such as sequence-dependent changeovers, unit release times, or discrete resource constraints. The basic idea is to use binary variables to represent the precedence relations between each pair of operations. The corresponding models usually involve many big-M constraints, which may result in poor LP relaxations. However, these formulations lead to models of modest size which often makes them tractable. In principle, these formulations can be used to represent any type of scheduling constraints, but global features such as limited inventory or discrete resource constraints, which may involve more than two operations, are non-trivial

and often require additional big-M constraints.

Another scheduling approach consists of using the concept of slots in order to assign a *position* for each operation in a *sequence*. Depending on the formulation used, these positions are directly or indirectly linked to the timing decisions. These types of formulations are particularly efficient to represent global scheduling features as its inherent sequencing representation may involve as many operations as needed.

Ku and Karimi (1988) presented an MILP formulation for multi-stage batch scheduling problem with finite interstage storage and exactly one unit per stage. Two heuristics are presented in order to solve the problem faster and compared to the optimal MILP approach.

Pinto and Grossmann (1995) developed a continuous-time slot-based mathematical formulation and considered a batch preordering heuristic to improve the computational requirements. This approach was applied to several examples with unit-dependent processing times and changeovers.

Hui et al. (2000) addressed the issue of sequence-dependent changeovers using a similar slot-based MILP model. Later, Gupta and Karimi (2003) improved the mathematical formulation using fewer variables and constraints to reduce the computational times.

Castro and Grossmann (2005) used the resource-task network representation (RTN) to model multi-stage batch scheduling problem. They developed and implemented several scheduling approaches and presented extensive computational results.

Recently, Prasad and Maravelias (2008) solved the integrated batching and scheduling problem. It consists of simultaneously making the following decisions: batch selection and sizing, unit assignment and sequencing of batches on each unit.

Finally, it should be noted that models for structures others than multi-stage batch plants have also been extensively studied using representations as the state-task network (Kondili et al., 1993) and resource-task network (Pantelides, 1994).

1.2 Optimization of Oil Refineries

Oil refineries are a key element in the valorization of crude-oils into energy. They consist of highly flexible plants that can refine crude-oils produced from many locations in the world, with very different properties, into useful petroleum products: LPG, gasoline and diesel fuels, kerosene, heating oil, bitumen, etc.

As depicted in Figure 1.2, the refining process can be decomposed into 3 main phases: crude-oil unloading and preparation for distillation, fractionation and reaction operations, final product blending and shipping. Based on this spatial decomposition, the following off-line optimization problems have been addressed in the literature:

1. Refinery planning (see section 1.2.1)
2. Crude-oil operations scheduling (see section 1.2.2)
3. Final product blending operations scheduling with recipe optimization (see Méndez et al., 2006b)
4. Scheduling of internal refinery product-specific subsystems:
   - for LPG, see Pinto and Moro (2000)
   - for fuel oils and asphalts, see Joly and Pinto (2003)
   - for lube oils and paraffins, see Casas-Liza and Pinto (2005)

Given the diversity and the complexity of the subsystems to be studied, this thesis is focused on the integration of the first two refinery optimization problems.

1.2.1 Refinery Planning

Refinery operators typically determine their annual and monthly plan by solving the refinery planning problem. It is based on a steady-state flowsheet optimization of a plant with multiple heterogenous units (see Fig. 1.3). The objective is to maximize the plant's net present value determined by sales revenues minus purchase and operating costs. In addition to process constraints, the problem also considers crude availabilities as well as product demands and specifications.

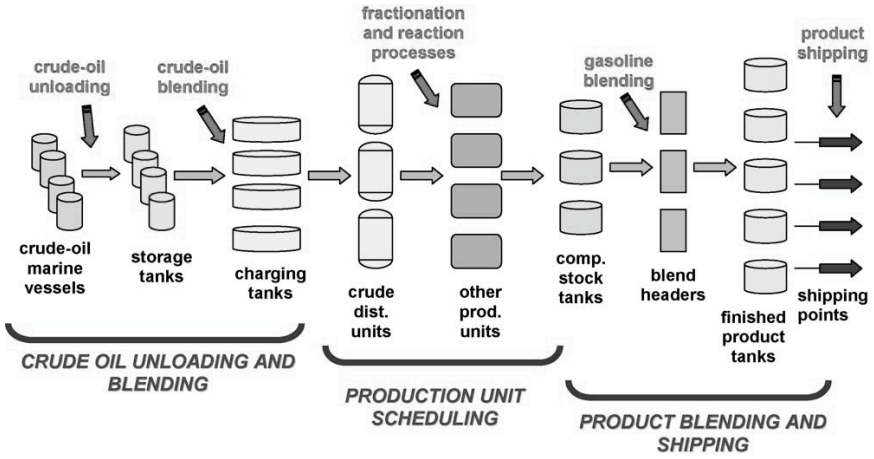


Figure 1.2: A typical oil refining process from Méndez et al. (2006b).

The refinery planning problem was one of the first industrial applications of linear programming (Bodington and Baker, 1990). However, the solution methods have evolved towards successive linear programming (SLP) in order to better account for the nonlinear nature of the refining process. In particular, the nonlinearities in the refinery model arise from *pooling equations* and *advanced process models*.

The pooling problem has received much attention in the literature since the 70's. It usually consists of optimizing feedstocks purchases, product blending operations and product sales while taking into account product availabilities, product demands, property specifications, and pool capacities. Haverly (1978); Hart (1978); Haverly (1980) developed and experimented the distributive recursion approach, a technique proved to be equivalent to classical SLP (see Lasdon and Joffe, 1990) in order to solve it.

Several authors have successively addressed the issue of global optimization of pure pooling problems. The techniques used range from generalized Benders decomposition (Floudas and Aggarwal, 1990), Lagrangian relaxation (Floudas and Visweswaran, 1993; Adhya et al., 1999), or spatial Branch and Bound (Foulds et al., 1992; Quesada and Grossmann, 1995a; Audet et al., 2000) to reformulation-linearization techniques (Audet et al., 2000; Meyer and Floudas, 2006), or piecewise convex relaxations (Meyer and Floudas, 2006; Pham et al.,

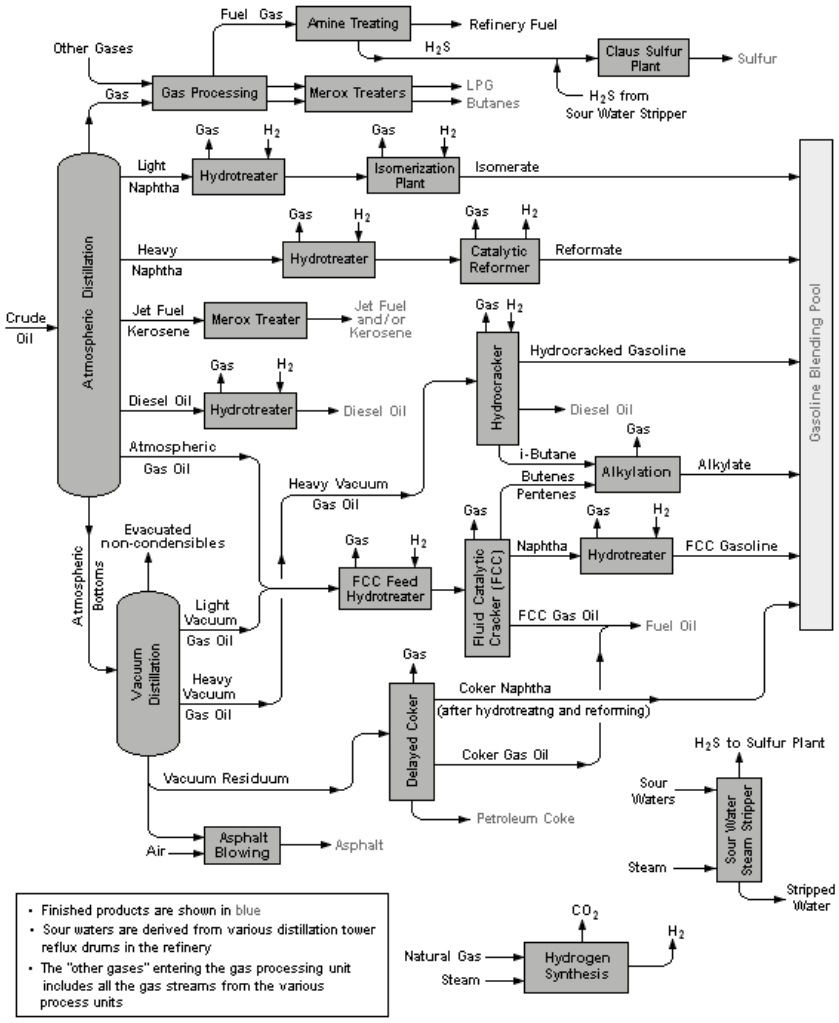


Figure 1.3: Schematic flow diagram of a typical oil refinery from Wikipedia (2010).

2009). Extensive reviews of pooling formulations and solution methods can be found in Audet et al. (2004); Misener and Floudas (2009).

Some examples of nonlinear refinery planning problems including pooling constraints and nonlinear process models can be found in Pinto and Moro (2000); Li et al. (2005); Alhajri et al. (2008). Although commercial solvers such as GRTMPS (Haverly Systems), PIMS (Aspen Tech), and RPMS (Honeywell Hi-Spec Solutions) implement successive linear programming algorithms to solve this problem (see Zhang et al., 1985), any standard NLP

solvers can also be used, although they may not guarantee global optimality of the solution.

A major issue with refinery planning is that most models are single-period models in which the refinery is assumed to operate in the same state over the whole planning period (typically 1 month). Therefore, the planning solution is used as a tactical goal for refinery operators rather than as an operational tool. In particular, crude distillation unit (CDU) feedstock decisions returned by the refinery planning problem are usually not applicable in the field due to crude logistics constraints. These are described in the crude-oil operations scheduling problem.

1.2.2 Crude-Oil Operations Scheduling

The optimal scheduling of crude-oil operations have been studied since the 90's and has been shown to lead to multimillion dollar benefits by Kelly and Mann (2003) as it is the first stage of the oil refining process. It involves crude-oil unloading from crude marine vessels (at berths or jetties), or from a pipeline to storage tanks, transfers from storage tanks to charging tanks and atmospheric distillations of crude-oil mixtures from charging tanks (see Fig. 1.4). The crude is then processed in order to produce basic products which are then blended into finished products (see section 1.2.1). In this thesis, we assume that the schedule of the crude supply is given. The production demands are determined by the long-term refinery planning, either sequentially (chapters 3, 4 and 5) or simultaneously (chapter 6). The following objectives are considered:

1. Maximization of crude gross margins (chapters 3 and 4)
2. Minimization of total logistics costs (chapter 5)
3. Maximization of total refinery profit as expressed in the planning problem (chapter 6)

Shah (1996) proposed to use mathematical programming techniques to find crude-oil schedules exploiting opportunities to increase economic benefits. Lee et al. (1996) considered a crude-oil scheduling problem involving crude unloading at berths, and developed a discrete-time MINLP model, and solved an MILP relaxation of the model. Later,

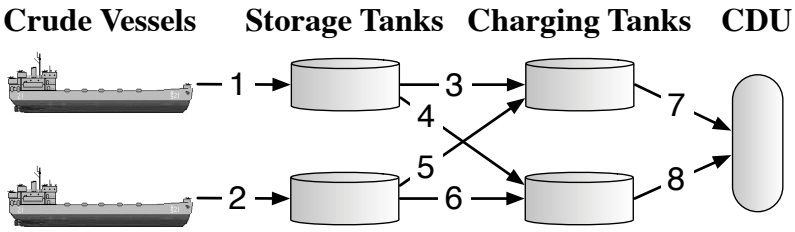


Figure 1.4: Crude-oil scheduling problem 1 from Lee et al. (1996).

Wenkay et al. (2002) improved the model and proposed an iterative approach to solve the MINLP model, taking into account the nonlinear blending constraints.

Pinto et al. (2000), Moro and Pinto (2004), and Reddy et al. (2004) used a global event formulation to model refinery systems involving crude-oil unloading from pipeline or jetties. The scheduling horizon is divided into fixed length sub-intervals, which are then divided in several variable length time-slots.

In parallel, Jia et al. (2003) developed an operation specific event model and applied it to the problems introduced by Lee et al. (1996) using a linear approximation of storage costs. A comparison of computational performance between both continuous-time and discrete-time models was given showing significant decreases in CPU time for the former model. Also, solutions that are not guaranteed to be globally optimal were obtained using standard MINLP algorithms.

Recently, Furman et al. (2007) presented a more accurate version of the event-point formulation, and Karuppiah et al. (2008) later addressed the global optimization of this model using an outer-approximation algorithm where the MILP master problem is solved by adding cuts from a Lagrangian decomposition. While rigorous, this method can be computationally expensive.

1.3 Mixed-Integer Optimization Tools

Scheduling problems are among the most challenging optimization problems, both in terms of modeling and solution algorithm. Mostly mixed-integer linear programming (MILP,

see Kallrath, 2002), constraint programming (CP, see Baptiste et al., 2001) and genetic algorithm (GA, see Mitchell, 1998) techniques have been used to tackle these problems. CP has proved to be very efficient for solving scheduling problems but it is rarely used to solve problems arising in the chemical engineering field. One of the reason is that CP is very efficient at sequencing tasks or jobs that are defined a priori (e.g. job-shop problems in discrete manufacturing). However, the scheduling of chemical processes usually involves both defining and sequencing the tasks that should be performed. Defining tasks means choosing a batch size or a unit operating mode for example. As a consequence, LP based techniques have been preferred with formulations essentially based on time grids as it easily allows modeling tank or unit capacity at the end of each time interval (see Floudas and Lin, 2004; Méndez et al., 2006b).

In this thesis, we aim at solving optimization problems involving both continuous and discrete decisions. Many computational techniques have emerged from the area of mixed-integer optimization in order to solve problems with different characteristics (linear, non-linear, convex, non-convex, purely integer, ...), including :

1. mixed-integer linear programming (MILP)
2. mixed-integer nonlinear programming (MINLP)
3. constraint programming (CP)
4. Lagrangian relaxation

In this section, we present brief reviews of these four optimization techniques as well as symmetry-breaking approaches.

1.3.1 Mixed-Integer Linear Programming

Mixed-integer linear programming is used to model many decision problems from industry (for example, process scheduling, production planning, resource allocation and supply chain management). However, solving such combinatorial models is NP-hard (see Nemhauser and Wolsey, 1999). Therefore, several approaches have been developed and combined in order

to solve these problems in reasonable times. Two main techniques have emerged: *branch-and-bound* (Land and Doig, 1960) and *cutting planes* (Gomory, 1958). Both are based on the *LP relaxation* of the MILP. This relaxation is obtained by considering integer variables as continuous variables with identical bounds: it can also be called *continuous relaxation* or *polytope relaxation*.

The branch-and-bound technique consists of searching through a tree defined by successive assignments of integer values to integer variables. At each node of this tree, an LP is solved in order to obtain a local node optimality bound (e.g. upper bound for maximization problems); the global optimality bound is the best bound among all open nodes (i.e. unprocessed nodes). If the solution of this LP is integral, it provides a global feasibility bound (e.g. lower bound for maximization problems). The search continues until all nodes have been processed (0% optimality gap) or until the optimality gap is below a specified tolerance.

The cutting plane algorithm consists of iteratively solving the LP relaxation and generating additional constraints (called cutting planes or cuts) that cut off the current LP solution. The procedure is stopped when the LP solution is integral. Although this can be achieved in a finite number of steps, in practice a large number of iterations are required.

The *branch-and-cut* procedure is a combination of the two aforementioned techniques. The cutting plane algorithm is used to tighten the LP relaxation at each node, thus improving the optimality bound (local and potentially global too). Branching occurs whenever the optimality bound cannot be significantly improved.

Recent reviews of MILP techniques can be found in Bixby et al. (1999); Johnson et al. (2000). The best known MILP solvers (CPLEX, Xpress, and Gurobi) all implement the branch-and-cut procedure and are widely used to solve industrial large-scale mixed-integer problems.

Table 1.1: Optimization techniques used in different MINLP solvers.

MINLP solvers	DICOPT	SBB and MINLP_BB	AlphaECP	Bonmin	KNITRO
branch-and-bound		x		x	x
outer-approximation	x			x	
LP/NLP based branch and bound				x	x
extended cutting plane			x		

1.3.2 Mixed-Integer Nonlinear Programming

A number of industrial applications of mixed-integer optimization include nonlinearities in the objective function or in the constraints. Except for specific cases (mixed-integer quadratic programming, mixed-integer quadratically constrained programming or mixed-integer second-order cone programming), standard MILP techniques cannot be used directly to solve such problems. Therefore, many optimization techniques have been developed to solve general MINLPs (for review, see Grossmann, 2002), including:

1. NLP-based branch-and-bound (Leyffer, 2001)
2. outer-approximation (Duran and Grossmann, 1986)
3. LP/NLP based branch and bound (Quesada and Grossmann, 1992)
4. extended cutting plane (Westerlund and Pettersson, 1995)

Table 1.1 summarizes the different techniques used in standard MINLP solvers available in GAMS.

A challenging MINLP topic is global optimization of non-convex problems, that is MINLPs with non-convex NLP subproblems. Similarly to MILP optimization, all approaches use a *convex relaxation* of the MINLP and rely on a spatial branch-and-bound search. These global optimization techniques include:

1. branch-and-reduce (Tawarmalani and Sahinidis, 2004)
2. α-BB (Adjiman et al., 2000)

3. spatial branch-and-bound for bilinear and linear fractional terms (Quesada and Grossmann, 1995a)
4. outer-approximation (Kesavan et al., 2004)

The main global MINLP solvers are BARON (Sahinidis, 1996), which implements an advanced branch-and-reduce algorithm, and LINDOGlobal (Lin and Schrage, 2009), which also implements a spatial branch-and-bound procedure. The global optimization of large-scale industrial non-convex MINLP problems is still unachievable in most cases. However, in some cases, recent developments in this area can be used to generate good heuristic solutions to such problems and provide tight global optimality estimates for these solutions.

1.3.3 Constraint Programming

Constraint programming is an alternative optimization approach to classical Operations Research (OR) techniques that is widely used to solve combinatorial problems such as scheduling (Baptiste et al., 2001), timetabling (Goltz and Matzke, 1998) or vehicle routing problems (Shaw, 1998). It is based on variable domain filtering algorithms, constraint propagation techniques and tree search heuristics. A central tool in constraint programming is the *domain store* (also called *constraint store*), which is used to record the domain of each variable in the model. At each node, the constraint propagation procedure iteratively calls each constraint domain filtering algorithm and updates the local domain store. If all variable domains are eventually reduced to singletons, a new solution if found, otherwise branching is used and a new node is selected. General reviews of constraint programming techniques can be found in van Hentenryck (1989); Rossi et al. (2006).

Many solvers implements these algorithms including Ilog CP, Choco, Gecode, and Comet (Michel and van Hentenryck, 2003). Recent developments in the CP community aim at improving filtering algorithms for global constraints (Régin, 2003; van Hoeve et al., 2009), integrating OR techniques with CP (Milano and Wallace, 2006; Hooker, 2007; Yunes et al., 2010), and replacing the traditional domain store by multi-valued decision diagrams (MDD)

to increase the amount of propagation between constraints (Andersen et al., 2007; Hoda et al., 2010).

The use of CP to solve optimization problems with continuous decisions, although theoretically achievable, has not yet received much attention and often relies on the discretization of the domain of the continuous variables. As a consequence, it is not widely used to solve industrial problems in the chemical industry. However, classical OR techniques can benefit from CP filtering algorithms and constraint propagation, which can be less computationally expensive than solving large strengthened LP models.

1.3.4 Lagrangian Relaxation

Lagrangian relaxation is a relaxation technique which aims at solving mathematical models including *complicating* or *hard* constraints. It consists of two major elements: a *primal relaxation* and a *dual algorithm*. The primal relaxation is obtained by transferring the complicating constraints into the objective function, scaled by a penalty factor, specifically the *Lagrange multiplier*. Given a Lagrange multipliers $\lambda \in \mathbb{R}_+^{m_2}$ (m_2 being the number of complicating constraints), this transformation can be described as follows:

$$(1) \begin{cases} \max & c^T x \\ \text{s.t.} & A_1 x \leq b_1 \\ & A_2 x \leq b_2 \end{cases} \Rightarrow (2) \begin{cases} \max & c^T x + \lambda^T (b_2 - A_2 x) \\ \text{s.t.} & A_1 x \leq b_1 \end{cases}$$

Problem (2) is a relaxation of (1) as its optimal objective value is always greater than the optimal objective value of (1).

Given this primal relaxation, the dual algorithm aims at solving the following problem:

$$(3) \min_{\lambda \in \mathbb{R}_+^{m_2}} \begin{cases} \max & c^T x + \lambda^T (b_2 - A_2 x) \\ \text{s.t.} & A_1 x \leq b_1 \end{cases}$$

In order to solve this dual problem, the following techniques can be used:

1. subgradient method (Held and Karp, 1971; Fisher, 1981)
2. cutting plane procedure (Cheney and Goldstein, 1959; Kelley, 1960)

3. boxstep method (Marsten et al., 1975)
4. bundle method (Lemaréchal, 1974)
5. volume algorithm (Barahona and Anbil, 2000)
6. analytic center cutting plane method (Goffin et al., 1992).

As explained in Frangioni (2005), problem (3) is equivalent to the following convexified version of the original problem:

$$(4) \begin{cases} \max & c^T x \\ \text{s.t.} & conv(A_1 x \leq b_1) \\ & A_2 x \leq b_2 \end{cases}$$

Also, the optimal Lagrange multipliers determined by solving problem (3) correspond to the marginal values of the constraint $A_2 x \leq b_2$ in an optimal solution of problem (4). Clearly, if all variables are continuous ($x \in \mathbb{R}^n$), problem (3) is therefore equivalent to problem (1). However, if some variables are integer, problem (3) yields a relaxation of problem (1). The difference between their respective optimal values is called a *dual gap*. In general, the Lagrangian relaxation is tighter than the LP relaxation, but it is more difficult to obtain. Although the Lagrangian relaxation technique have been developed for LPs or MILPs, it can also be used to solve difficult MINLPs (for example, see Neiro and Pinto, 2006).

A key issue in Lagrangian relaxation techniques is the choice of the complicating constraints. There is a classic tradeoff between making the relaxed problem (2) easy to solve and reducing the dual gap. In some cases, the complicating constraints are selected in order to make the relaxed problem (2) decomposable and therefore much easier to solve. This technique is called *Lagrangian decomposition*. The reader may refer to Fisher (1985) and Guignard (2003) for extensive reviews on Lagrangian relaxation and decomposition techniques.

1.3.5 Symmetry-Breaking Approaches

The effectiveness of the optimization techniques previously mentioned often suffers from the presence of multiple symmetric solutions. In simple words, we define symmetric solutions as feasible solutions that provide identical practical decisions from the modeler's perspective. In particular, symmetric solutions have identical objective values. The presence of symmetries usually leads to an exhaustive enumeration of many feasible solutions which are not detected as symmetric by the solver. An extensive review of symmetry detection and symmetry-breaking approaches can be found in Margot (2008). In the context of this thesis, we focus on problem-specific symmetry detection and static symmetry-breaking constraints.

1.4 Overview of Thesis

1.4.1 Chapter 2

In chapter 2, a unified modeling approach for solving process scheduling problems is proposed. Four different time representations that are based on priority-slots are presented and compared by deriving the relations between them. For each time representation, a mathematical model is presented and strengthened using the maximum cliques and bicliques of the non-overlapping graph. We introduce three solution methods that can be used to achieve global optimality or obtain near-optimal solutions depending on the stopping criterion used. The proposed approaches are applied to single-stage and multi-stage batch scheduling problems. Computational results show that the multi-operation sequencing (MOS) time representation is superior to the others as it allows efficient symmetry-breaking and requires fewer priority-slots, thus leading to smaller model sizes.

1.4.2 Chapter 3

In chapter 3, the crude-oil operations scheduling problem is stated and the four time representations introduced in chapter 2 are used to derive strengthened MINLP models for

solving this problem. In particular, it is shown how the non-overlapping graph can be used to generate non-trivial strengthened constraints for a specific example. Computational results are obtained for all time representations except for single-operation sequencing (SOS). A two-step MILP-NLP procedure is used to solve the non-convex MINLP models leading to an optimality gap lower than 4% in all cases.

1.4.3 Chapter 4

In chapter 4, we apply the single-operation sequencing (SOS) time representation introduced in chapter 2 to the crude-oil scheduling problems presented in chapter 3. The corresponding MINLP model is based on the representation of a crude-oil schedule by a single sequence of transfer operations. Therefore, it is possible to reduce the symmetries involved in the problem using a deterministic finite automaton to represent a symmetry-breaking sequencing rule. Computational results show the effectiveness of the symmetry-breaking approach. A final comparison of all time representations applied to the crude-oil scheduling operations problem is presented showing the superiority of the MOS model.

1.4.4 Chapter 5

Chapter 5 aims at tightening the linear relaxation of a refinery crude-oil operation scheduling MINLP based on the single-operations sequencing (SOS) time representation. The model is mostly linear but contains bilinear products of continuous variables in the objective function (minimization of total logistics costs). It is possible to define a linear relaxation of the model leading to a weak bound on the objective value of the optimal solution. A typical method to address this issue is to discretize the continuous space and to use linear relaxation constraints based on lower and upper bounds of the variables (e.g. McCormick convex envelopes, see McCormick, 1976) on each subdivision of the continuous space. This work explores another approach involving constraint programming (CP). The idea is to use an additional CP model which is used to tighten the bounds of the continuous variables involved in bilinear terms and then generate cuts based on McCormick convex envelopes.

These cuts are then added to the mixed-integer linear program (MILP) during the search leading to a tighter linear relaxation of the MINLP. Results show large reductions of the optimality gap in the two-step MILP-NLP solution method introduced in chapter 3 due to the tighter linear relaxation obtained.

1.4.5 Chapter 6

In chapter 6, we introduce a new methodology to solve a large-scale mixed-integer nonlinear program (MINLP) integrating the two major optimization problems appearing in the oil refining industry: refinery planning and crude-oil operations scheduling. The proposed approach consists of using Lagrangian decomposition to effectively integrate both problems. The main advantage of this technique is that each problem can be solved independently. A new hybrid dual problem is introduced to iteratively update the Lagrange multipliers. It uses the classical concepts of cutting planes, subgradient, and boxstep. The proposed approach is compared to a basic sequential approach and to standard MINLP solvers. The results obtained on a case study and a larger refinery problem show that the Lagrangian decomposition algorithm is more robust than the other approaches and produces better solutions in reasonable times.

1.4.6 Chapter 7

Chapter 7 summarizes the major contributions of the thesis. The conclusions are discussed with suggestions for future work.

This thesis has led to the following papers: Mouret et al. (2008, 2009a,b, 2010a,b)

Chapter 2

Time Representations and Mathematical Models for Process Scheduling Problems

2.1 Introduction

Rigorous optimization of real-world problems are often based on advanced optimization tools such as mixed-integer linear programming (MILP) or constraint programming (CP, see Rossi et al., 2006). These tools rely on a mathematical or symbolic representation of the problem that is applied by an end-user. In some cases, the relationship between the problem description and its mathematical model is not clear. Therefore an intermediate step is included in the optimization approach (see Figure 2.1). In this step, the representation used is detailed and approximations are made. For instance, in the context of scheduling problems, using a discrete-time formulation is in general a constraining approximation of the actual problem, and thus, it may lead to a suboptimal solution as discussed in Floudas and Lin (2004).

Additionally, it is important to note that several mathematical models may be used to obtain the global optimal solution of the problem, which is the best possible solution according to a given optimization criterion. For example, many time representations rely on a specific parameter representing the number of time points (Kondili et al., 1993), time intervals (Lee et al., 1996), or event points (Ierapetritou and Floudas, 1998). Therefore, the scheduling problem is represented by an infinite set of mathematical models, one for each possible value of this parameter (all positive integers). The global optimal schedule is the best solution among the optimal solution of all these models. In general, it is not possible

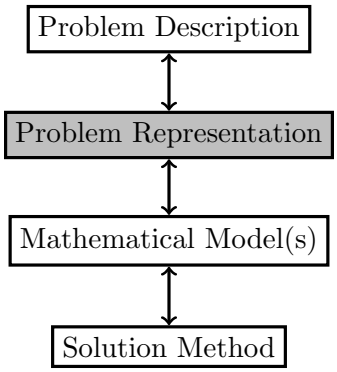


Figure 2.1: Four steps optimization approach.

to know a priori the parameter value that will lead to the global optimal solution, although it is sometimes possible to derive upper and lower bounds for it. The common trade-off is that global optimality may be guaranteed with a large value of this parameter, which often results in prohibitive solution times.

Many different time representations have been introduced to solve scheduling problems (for review see Floudas and Lin, 2004). Experience has shown that, depending on the characteristics of the problem, some time representations are more suitable than others. In this thesis, we focus on scheduling problems that rely on:

a) a set of possible *operations*, or actions, that can be performed once, several times, or not at all;
b) *scheduling decisions* that involve both selecting, parametrizing and sequencing the operations that should be executed;
c) *scheduling constraints* such as release dates, due dates, bounds on processing times, non-overlapping constraints, sequence-dependent changeovers, cardinality constraints, and precedence constraints;
d) additional *side constraints* that are used to model more complex features such as limited inventory management or process constraints.

It should be noted that, for instance, the selection of operations may correspond to the selection of equipment or discrete resources for tasks in a state-task-network (Kondili et al.,

1993) or in a resource-task-network (Pantelides, 1994). In general, operations are defined by fully disaggregating all possible discrete selections of actions in the scheduling system. In contrast, parameterization of operations corresponds to continuous decisions such as batch sizes, transfer volumes, or process operating conditions. The above assumptions do not cover all kinds of scheduling problems but are an important part of the classification presented by Méndez et al. (2006a). A unique aspect of this work is that the unifying framework of the models presented in this chapter allows it to be applied to single-stage and multi-stage batch scheduling problems as well as to crude-oil operations scheduling (see chapters 3 and 4).

The main objective of this chapter is to develop a unified modeling approach for scheduling problems in order to facilitate the evaluation of several time representations, both in terms of computational time and solution quality. First, a simple scheduling problem is introduced as an example. Next, we study four different types of time representations, which have been used in the literature and clarify the relationships between them. Then, basic MILP models for pure scheduling constraints are presented for each of these time representation. Using concepts from graph theory (cliques and bicliques), we show how these models can be generically strengthened based on the structure of the scheduling problem. Three solution methods are then developed to solve these mathematical formulations. Finally, single-stage and multi-stage batch scheduling problems are presented and solved using the different approaches in order to show the effectiveness of the strengthened formulations, and to provide elements of comparison between the different time representations.

2.2 Case Study

We introduce a small scheduling system that involves 6 different operations $v_1, \ldots, v_6$ and 3 unary resources r_1, r_2, r_3. A unary resource cannot be shared by two or more processing operations at a given time. Table 2.1 displays resource requirement for each operation. For example, operation v_4 simultaneously requires resources r_1 and r_2. In this case and

Table 2.1: Resource requirements for case study.

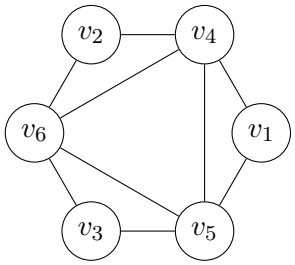
Operation	v_1	v_2	v_3	v_4	v_5	v_6
Resources	r_1	r_2	r_3	$r_1 \wedge r_2$	$r_1 \wedge r_3$	$r_2 \wedge r_3$

in the examples studied in this thesis, unary resource requirements are handled as non-overlapping constraints between operations. For instance, operations v_1 and v_4 cannot overlap as they both use resource r_1. Also, operations v_5 and v_6 cannot overlap as they both use resource r_3. Besides, as a given operation v can be executed several times, any two separate executions of v may not overlap. Thus, any operation v cannot overlap with itself. Different linear objectives can be considered: maximization of profit, minimization of makespan, minimization of assignment costs, minimization of tardiness or earliness.

In order to extract useful information from the structure of the problem, we use a global representation of all the non-overlapping constraints. The *non-overlapping matrix*, denoted by NO, is such that $NO_{vv'} = 1$ if operation v and v' must not overlap, 0 otherwise. The *non-overlapping graph*, denoted by $G_{NO} = (W, E)$, is an undirected graph where the set of vertices W is the set of operations and the set of edges is defined by $E = \{\{v, v'\} \text{ s.t. } NO_{vv'} = 1\}$. Therefore, the non-overlapping matrix is the adjacency matrix of graph G_{NO}. The concept of non-overlapping graph can be viewed as an extension of the disjunctive graph (Balas, 1969; Adams et al., 1988), which is used to represent disjunctive constraints between operations that have to be executed exactly once. In this thesis, we consider operations that can be executed once, several times, or not at all. Figure 2.2 shows the non-overlapping matrix and graph for the case study. For clarity, edges that connect a vertex to itself, called self-loops, are not represented.

2.3 Time Representations

In this section, we study four different time representations and show how they can be defined using identical concepts. Each representation makes use of a totally ordered set of priority-slots $T = \{1, \ldots, n\}$, which are used to assign and order the executions of op-



$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Figure 2.2: Non-overlapping matrix and graph for case study.

erations. The number of priority-slots, denoted by n, has to be postulated a priori. Any operation may be executed several times by assigning it to multiple priority-slots. We denote by $A = \{(i, v), i \in T, v \in W\}$ the set of all possible assignments and define a partial scheduling order on A by:

$$\forall (i_1, v_1), (i_2, v_2) \in A, \quad (i_1, v_1) \prec (i_2, v_2) \Leftrightarrow i_1 < i_2 \wedge NO_{v_1 v_2} = 1$$

In each time representation, a solution is defined as a subset A' of A and is represented by a sequence of operations. Each selected assignment (i, v) correspond to an execution of operation v with scheduling priority i during time interval $[S_{iv}, E_{iv}]$. In each time representation, the partial scheduling order $\prec$ implies precedence relations between elements of A' as follows:

$$\forall (i_1, v_1), (i_2, v_2) \in A', \quad (i_1, v_1) \prec (i_2, v_2) \Rightarrow E_{i_1 v_1} \leq S_{i_2 v_2}$$

It is should be noted that it is not straightforward to select the number of priority-slots. Indeed, postulating a large number priority-slots increases the chance of obtaining the global optimal solution, but it also increases the size of the model and the CPU time. The four time representations are listed below.

a. Multi Operation Sequencing (MOS)
b. Multi Operation Sequencing with Synchronized Start Times (MOS-SST)
c. Multi Operation Sequencing with Fixed Start Times (MOS-FST)
d. Single Operation Sequencing (SOS)

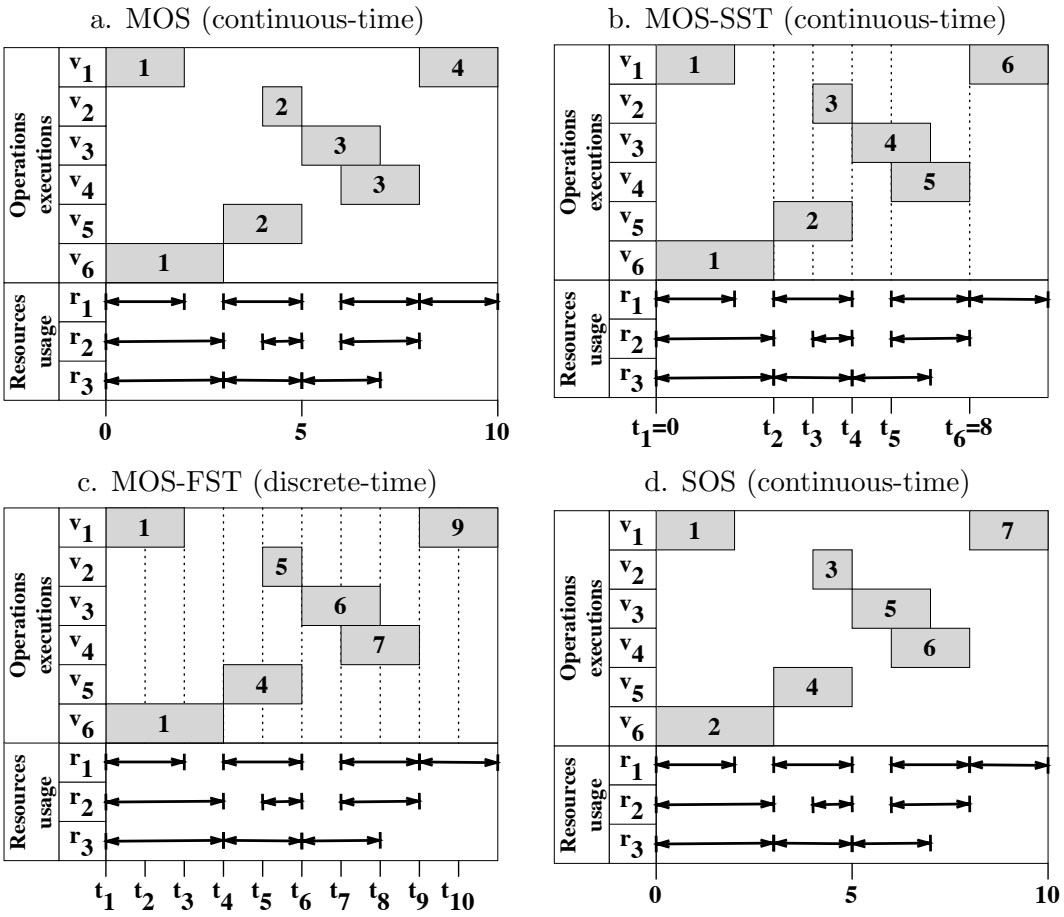


Figure 2.3: A unique schedule obtained through different time representations.

Figure 2.3 shows how the same schedule for the case study can be obtained within each time representations. Each execution of an operation is represented by an horizontal bar in the upper Gantt chart, while resource usage is represented by horizontal lines in the lower Gantt chart. The priority-slots are represented by number labels on each operation execution. In each case, the smallest possible number of priority-slots needed to obtained the solution has been used. From this figure, it is clear that some time representations require more priority-slots than others.

In the **MOS** representation, several operations can be assigned to each priority-slots as long as they may overlap with each other. For instance, in Figure 2.3(a) operations v_1 and v_6

are allowed to overlap and are both assigned to the first priority-slot. However, operations v_1 and v_5 cannot overlap and are consequently assigned to different priority-slots: slots 1 and 4 for operation v_1, slot 2 for operation v_5. If two non-overlapping operations v and v' are assigned to priority-slots i and j, respectively, such that $i < j$, then operation v' must be executed after operation v (i.e. operation v must start after the end of operation v). For instance, operation v_1 assigned to priority-slot 4 is executed after operation v_2 assigned to priority-slot 2. The solution depicted in Figure 2.3(a) is represented by the sequence of operations $(\{\mathbf{1},\mathbf{6}\},\{\mathbf{2},\mathbf{5}\},\{\mathbf{3},\mathbf{4}\},\{\mathbf{1}\})$. We denote **MOS($n$)** a scheduling model using the MOS time representation with n postulated priority-slots. This time representation was introduced by Ierapetritou and Floudas (1998) as the event point formulation. Their mathematical model, although significantly different than the model developed in this thesis, was used to solve several STN problems. As mentioned by Maravelias and Grossmann (2003), inventory tracking using event points is quite different than inventory tracking using time points, which might lead to inconsistent enforcement of storage capacity constraints. This issue was addressed by Janak et al. (2004) by adding additional storage tasks in the STN problem, which can lead to a significant increase of model size.

The **MOS-SST** representation is based on the same features as the MOS representation. Additionally, all operations assigned to the same priority-slot must have the same start time. For instance, in Figure 2.3(b), operations v_1 and v_6 are both assigned to priority-slot 1, and therefore both start at the same time $t = 0$. Thus, each priority-slot i is associated to variable time-point t_i which is represented by a vertical dotted line in Figure 2.3(b). The time interval between any two successive time-points is variable. The solution depicted in Figure 2.3(a) is represented by the sequence of operations $(\{\mathbf{1},\mathbf{6}\},\{\mathbf{2}\},\{\mathbf{5}\},\{\mathbf{3}\},\{\mathbf{4}\},\{\mathbf{1}\})$. We denote **MOS-SST($n$)** a scheduling model using the MOS-SST time representation with n postulated priority-slots. This type of representation has been used to solve a wide variety of problems where time-points are used to track both the start and end events of each operation (see Zhang and Sargent, 1996; Schilling and Pantelides, 1996; Maravelias and Grossmann, 2003).

The **MOS-FST** representation is based on the same features as the MOS-SST representation. Additionally, the time-point associated to each priority-slot is fixed a priori. Thus, the interval between any two successive time-points is fixed. For instance, the solution depicted in Figure 2.3(c) is obtained using time-points that are uniformly spaced along the time horizon: $t_1 = 0, t_2 = 1, \ldots, t_{10} = 9$. Therefore, operation v_5 assigned to priority-slot 4 starts at $t = t_4 = 3$ while operation v_4 assigned to priority-slot 7 starts at $t = t_7 = 6$. The solution depicted in Figure 2.3(c) is represented by the sequence of operations $(\{\mathbf{1},\mathbf{6}\}, \emptyset, \emptyset, \{\mathbf{5}\}, \{\mathbf{2}\}, \{\mathbf{3}\}, \{\mathbf{4}\}, \emptyset, \{\mathbf{1}\}, \emptyset)$. We denote **MOS-FST($n$)** a scheduling model using the MOS-FST time representation with n postulated priority-slots. Discrete-time formulation for process scheduling problems were initially developed to solve STN and RTN models where processing times are assumed to be constant (see Kondili et al., 1993; Pantelides, 1994).

In the **SOS** representation, at most one operation can be assigned to each priority-slot. Similarly to the MOS model, if two non-overlapping operations v and v' are assigned to priority-slots i and j $(i < j)$, then v' must be executed after v. It should be noted that this constraint does not apply to pairs of operations that are allowed to overlap. As operations v_2 and v_5 are allowed to overlap, their relative position in time is not affected by their respective scheduling priority. Therefore, the proposed solution is equivalent to assigning operations v_2 and v_5 to priority-slots 4 and 3, respectively. The solution depicted in Figure 2.3(d) is represented by the sequence of operations $(\mathbf{1},\mathbf{6},\mathbf{2},\mathbf{5},\mathbf{3},\mathbf{4},\mathbf{1})$. We denote **SOS($n$)** a scheduling model using the SOS time representation with n postulated priority-slots. This time representation was introduced by Mouret et al. (2009a) to solve the refinery crude-oil operations scheduling problem.

Table 2.2 summarizes the correspondence between our nomenclature and equivalent naming conventions used in the literature. From these definitions, it can be inferred that for a given number of priority-slots n the integer feasible space of **MOS(n)** is larger than the integer feasible space of models **MOS-SST(n)**, **MOS-FST(n)**, and **SOS(n)**. Indeed, the latter models are derived from the MOS model by introducing additional constraints,

Table 2.2: Time representations nomenclature.

Nomenclature	Equivalent names
MOS	unit-specific time grid or multiple time grid
MOS-SST	global events or common time grid or nonuniform discretization
MOS-FST	fixed events or fixed time grid or uniform discretization
SOS	first introduced in Mouret et al. (2009a)

which reduce the set of feasible solutions. Furthermore, the integer feasible space of **MOS-SST**(n) is larger than the one of **MOS-FST**(n) and **SOS**(n). In particular, any solution for the **SOS** model is a solution for the **MOS-SST** model. Indeed, at most one operation can be assigned to each priority-slot so the synchronization of start times is automatically satisfied.

These properties can also be interpreted by considering a scheduling solution z. We denote $z \in \mathbf{MOS}(n)$ the membership of schedule z to the integer feasible space of model **MOS**(n). In other words, $z \in \mathbf{MOS}(n)$ means that solution z satisfies all the constraints of model **MOS**(n). We introduce the minimum number of priority-slots needed to "find" solution z using each time representation.

$$n_{\mathbf{MOS}}(z) = \min_n \{n | z \in \mathbf{MOS}(n)\}$$

$$n_{\mathbf{MOS\text{-}SST}}(z) = \min_n \{n | z \in \mathbf{MOS\text{-}SST}(n)\}$$

$$n_{\mathbf{MOS\text{-}FST}}(z) = \min_n \{n | z \in \mathbf{MOS\text{-}FST}(n)\}$$

$$n_{\mathbf{SOS}}(z) = \min_n \{n | z \in \mathbf{SOS}(n)\}$$

Then, the following inequalities hold:

$$\begin{cases} n_{\mathbf{MOS}}(z) \leq n_{\mathbf{MOS\text{-}SST}}(z) \leq n_{\mathbf{MOS\text{-}FST}}(z) \\ n_{\mathbf{MOS}}(z) \leq n_{\mathbf{MOS\text{-}SST}}(z) \leq n_{\mathbf{SOS}}(z) \end{cases}$$

The relation between MOS-SST and MOS-FST time representations was previously derived by Maravelias and Grossmann (2006) in the context of state-task network formulations.

Remark. An important limitation of these time representations is that operations are considered as a whole for sequencing purpose. More precisely, a scheduling priority is

assigned to operations and not to the start and end events of these operations, which may be necessary to solve some scheduling problems. For instance, operations that require a cumulative resource with capacity greater than 1 (e.g. manpower limited to 2 workers) are allowed to overlap, but only a limited number of these operations may overlap at any given point in time. The models developed in this thesis do not accommodate these features. Another case is inventory tracking when simultaneous charging and discharging of tanks is allowed. It is sufficient to enforce capacity limitations only at the start and end events of each charging/discharging operations, but in order to do so, a precise sequence of such events needs to be obtained by the model. Possible extensions of the model to handle these specific features, such as presented in Janak et al. (2004), will not be discussed in this thesis.

2.4 Mathematical Models

In this section, we present mathematical models for each time representation. They all rely on the same sets, parameters and variables. Objective functions are not presented here (e.g. minimize makespan, minimize tardiness, or maximize profit) although they can significantly impact the solution of the corresponding formulation.

2.4.1 Sets and Parameters

The following sets and parameters are used.

- $T = \{1, \ldots, n\}$ is a totally ordered set of priority-slots (indices i, j, i_1, i_2).
- W is the set of all operations (indices v, v', v_1, v_2).
- H is the scheduling horizon.
- $[\underline{S_v}, \overline{S_v}] \subset [0, H]$ are bounds on the start time of any execution of operation v.
- $[\underline{D_v}, \overline{D_v}] \subset [0, H]$ are bounds on the duration of any execution of operation v.
- $[\underline{E_v}, \overline{E_v}] \subset [0, H]$ are bounds on the end time of any execution of operation v.
- $[\underline{N_v}, \overline{N_v}]$ are bounds on the total number of executions of operation v.
- $[\underline{N_{W'}}, \overline{N_{W'}}]$ are bounds on the total number of executions of all operations in $W' \subset W$.

- $NO_{v_1v_2}$ is 1 if operations v_1 and v_2 must not overlap, 0 if they are allowed to overlap.
- $TR_{v_1v_2}$ is a sequence-dependent transition time between non-overlapping operations v_1 and v_2.
- $TR_{W'}$ is a unique set transition time between any pair of non-overlapping operations in $W' \subset W$.
- $P_{v_1v_2} = 1$ denotes a precedence constraint between operations v_1 and v_2.
- $P_{W_1W_2} = 1$ denotes a precedence constraint between sets of operations W_1 and W_2.

Remark 1: It should be noted that for operations with fixed processing time, $\underline{D_v} = \overline{D_v} = D_v$.

Remark 2: A set transition time $TR_{W'}$ is defined when $\forall v_1, v_2 \in W', TR_{v_1,v_2} = TR_{W'}$. It can be used to represent unit changeover times.

Remark 3: A precedence constraint between operations v_1 and v_2 states that v_1 must be executed before v_2. We assume that precedence constraints are enforced on operations which are executed exactly once ($\underline{N_{v_1}} = \underline{N_{v_2}} = \overline{N_{v_1}} = \overline{N_{v_2}} = 1$). Similarly, we consider that a precedence constraint between sets of operations W_1 and W_2 states that exactly one operation in each set must be executed ($\underline{N_{W_1}} = \underline{N_{W_2}} = \overline{N_{W_1}} = \overline{N_{W_2}} = 1$) and the operation selected from W_1 must be executed before the operation selected from W_2.

2.4.2 Variables

The variables used in all models are composed of binary assignment variables, and continuous time variables.

- **Assignment variables** $Z_{iv} \in \{0, 1\} \qquad i \in T, v \in W$

  $Z_{iv} = 1$ if operation v is assigned to priority-slot i, $Z_{iv} = 0$ otherwise.
- **Time variables** $S_{iv} \geq 0, D_{iv} \geq 0, E_{iv} \geq 0 \qquad i \in T, v \in W$

  S_{iv} is the start time of operation v if it is assigned to priority-slot i, $S_{iv} = 0$ otherwise.

  D_{iv} is the duration of operation v if it is assigned to priority-slot i, $D_{iv} = 0$ otherwise.

  E_{iv} is the end time of operation v if it is assigned to priority-slot i, $E_{iv} = 0$ otherwise.

2.4.3 MOS Model

Variable Bound Constraints Bounds on time variables can be expressed using the following constraints.

$$\underline{S_v} \cdot Z_{iv} \leq S_{iv} \leq \overline{S_v} \cdot Z_{iv} \qquad i \in T, v \in W \tag{2.1a}$$

$$\underline{D_v} \cdot Z_{iv} \leq D_{iv} \leq \overline{D_v} \cdot Z_{iv} \qquad i \in T, v \in W \tag{2.1b}$$

$$\underline{E_v} \cdot Z_{iv} \leq E_{iv} \leq \overline{E_v} \cdot Z_{iv} \qquad i \in T, v \in W \tag{2.1c}$$

Time Constraint Time variables are linked through the following additional constraint.

$$E_{iv} = S_{iv} + D_{iv} \qquad i \in T, v \in W \tag{2.2}$$

Cardinality constraint The total number of execution of operations in a set $W' \subset W$ is restricted by the following constraint. A cardinality constraint on a single operation v can be enforced by setting $W' = \{v\}$.

$$\underline{N_{W'}} \leq \sum_{\substack{i \in T \\ v \in W'}} Z_{iv} \leq \overline{N_{W'}} \qquad W' \subset W \tag{2.3}$$

Assignment Constraint Two non-overlapping operations v_1 and v_2 such that $NO_{v_1 v_2} = 1$ cannot be assigned simultaneously to the same priority-slot.

$$Z_{iv_1} + Z_{iv_2} \leq 1 \qquad i \in T, v_1, v_2 \in W, NO_{v_1 v_2} = 1 \tag{2.4}$$

Non-overlapping Constraint A non-overlapping constraint between two operations $v_1, v_2 \in W$ states that they must not be executed simultaneously. This property is enforced using the following big-M constraints where the big-M constant is defined as a valid upper bound of the left hand side of the inequality.

$$E_{i_1 v_1} \leq S_{i_2 v_2} + H \cdot (1 - Z_{i_2 v_2}) \qquad i_1, i_2 \in T, i_1 < i_2, v_1, v_2 \in W, NO_{v_1 v_2} = 1 \tag{2.5a}$$

$$E_{i_1 v_2} \leq S_{i_2 v_1} + H \cdot (1 - Z_{i_2 v_1}) \qquad i_1, i_2 \in T, i_1 < i_2, v_1, v_2 \in W, NO_{v_1 v_2} = 1 \tag{2.5b}$$

Due to the assignment constraint (2.4) and variable bound constraints (2.1a) and (2.1c), equations (2.5a) and (2.5b) can be combined in order to form the following tighter surrogate constraint (see appendix A).

$$\begin{aligned} &E_{i_1v_1} + E_{i_1v_2} \\ &\leq S_{i_2v_1} + S_{i_2v_2} + H \cdot (1 - Z_{i_2v_1} - Z_{i_2v_2}) \end{aligned} \qquad i_1, i_2 \in T, i_1 < i_2, v_1, v_2 \in W, NO_{v_1v_2} = 1 \quad (2.6)$$

Non-overlapping Constraint with Transition Times Sequence-dependent transition times $TR_{v_1v_2}$ between operations $v_1 \in W$ and $v_2 \in W$ can be enforced as follows.

$$\begin{aligned} &E_{i_1v_1} + TR_{v_1v_2} \cdot Z_{i_1v_1} \\ &\leq S_{i_2v_2} + (H + TR_{v_1v_2}) \cdot (1 - Z_{i_2v_2}) \end{aligned} \qquad i_1, i_2 \in T, i_1 < i_2, v_1, v_2 \in W, NO_{v_1v_2} = 1 \quad (2.7a)$$

$$\begin{aligned} &E_{i_1v_2} + TR_{v_2v_1} \cdot Z_{i_1v_2} \\ &\leq S_{i_2v_1} + (H + TR_{v_2v_1}) \cdot (1 - Z_{i_2v_1}) \end{aligned} \qquad i_1, i_2 \in T, i_1 < i_2, v_1, v_2 \in W, NO_{v_1v_2} = 1 \quad (2.7b)$$

If the transition time is not sequence-dependent (i.e. $TR_{v_1v_2} = TR_{v_2v_1}$) then constraints (2.7) can be combined in order to form the following tighter surrogate constraint.

$$\begin{aligned} &E_{i_1v_1} + E_{i_1v_2} + TR_{v_1v_2} \cdot (Z_{i_1v_1} + Z_{i_1v_2}) && i_1, i_2 \in T, i_1 < i_2, v_1, v_2 \in W, \\ &\leq S_{i_2v_1} + S_{i_2v_2} + (H + TR_{v_1v_2}) \cdot (1 - Z_{i_2v_1} - Z_{i_2v_2}) && NO_{v_1v_2} = 1, TR_{v_1v_2} = TR_{v_2v_1} \end{aligned} \quad (2.8)$$

Precedence Constraint Consider two operations $v_1, v_2 \in W$ such that $P_{v_1v_2} = 1$, then the precedence constraint can be expressed as follows. Note that in each summation, exactly one term can be non-zero due to the assumed cardinality constraint.

$$\sum_{i \in T} E_{iv_1} \leq \sum_{i \in T} S_{iv_2} \qquad v_1, v_2 \in W, P_{v_1v_2} = 1 \quad (2.9)$$

Consider two sets of operations $W_1, W_2 \subset W$. If $P_{W_1W_2} = 1$, the previous constraint can be extended as follows. Note that in each summation, exactly one term can be non-zero due to the corresponding cardinality constraint (2.3).

$$\sum_{i \in T} \sum_{v_1 \in W_1} E_{iv_1} \leq \sum_{i \in T} \sum_{v_2 \in W_2} S_{iv_2} \qquad W_1, W_2 \subset W, P_{W_1W_2} = 1 \quad (2.10)$$

2.4.4 MOS-SST Model

In the MOS-SST model, the start times of all operations assigned to the same priority-slot i have to be synchronized. Therefore, we introduce positive synchronization time-points variables t_i ($i \in T$). Variables t_i correspond to the start time of all operations assigned to priority-slot i.

$$t_i \in [0, H] \qquad i \in T \tag{2.11}$$

Time-point Sequence Constraint The following constraint ensures that the variables t_i are ordered in time.

$$t_{i-1} \leq t_i \qquad i \in T \setminus \{1\} \tag{2.12}$$

Synchronization Constraints If operation v is assigned to priority-slot i, it must start at time t_i. Therefore, the following synchronization constraints are used. Note that constraint (2.1a) already ensures that $S_{iv} = 0$ if $Z_{iv} = 0$.

$$S_{iv} \leq t_i \qquad i \in T, v \in W \tag{2.13a}$$

$$S_{iv} \geq t_i - H \cdot (1 - Z_{iv}) \qquad i \in T, v \in W \tag{2.13b}$$

2.4.5 MOS-FST Model

In the MOS-FST model, the time-points variables t_i of the MOS-SST model are used and are fixed a priori. They can therefore be considered as parameters. In this thesis, these time-points are always selected using a uniform time discretization.

$$t_i = \frac{i-1}{n} \cdot H \qquad i \in T \tag{2.14}$$

Synchronization Constraint If operation v is assigned to priority-slot i, it must start at time t_i. Therefore, the following synchronization constraints are used.

$$S_{iv} = t_i \cdot Z_{iv} \qquad i \in T, v \in W \tag{2.15}$$

2.4.6 SOS Model

All constraints from the MOS model are still valid for the SOS model. However, a new assignment constraint is defined.

Assignment Constraint At most one operation has to be assigned to each priority-slot.

$$\sum_{v \in W} Z_{iv} \leq 1 \qquad i \in T \tag{2.16}$$

2.5 Strengthened Reformulations

The models presented in section 2.4 may not be effectively solved by MILP solvers. Indeed, special attention needs to be paid to their LP relaxation. As MILP solvers usually perform better when the model has a tight LP relaxation, we introduce strengthened formulations based on the non-overlapping structure of the problem.

2.5.1 Non-overlapping Graph Properties

We recall the definition of a *clique* and a *biclique* in the context of the non-overlapping graph G_{NO}. A *clique* of G_{NO} is a subset of the set of operations $W' \subset W$ such that any two operations in W' must not overlap. A *maximal clique* is a clique that is not a subset of any other clique. An *isolated clique* is a clique that has no edges connecting it to its complement in the graph. For the case study, the non-overlapping graph displayed in Figure 2.2 contains 9 non-maximal cliques of two vertices, one for each non self-loop edge (e.g. $\{v_1, v_4\}$). It contains 4 maximal cliques of three vertices $\{v_1, v_4, v_5\}$, $\{v_2, v_4, v_6\}$, $\{v_3, v_5, v_6\}$, $\{v_4, v_5, v_6\}$. It contains no clique of larger size and no isolated clique.

We define a *biclique* of G_{NO} as a pair of sets of operations $(W_1; W_2) \in W^2$ such that for any pair of operations $(v_1; v_2) \in W_1 \times W_2$, $\{v_1, v_2\}$ is an edge of G_{NO}. Sets W_1 and W_2 are not necessarily disjoint. A *maximal biclique* of G_{NO} is a biclique that is not contained in any other biclique of G_{NO}. For the case study, the non-overlapping graph displayed in Figure 2.2 contains 9 non-maximal bicliques of two vertices, one for each non self-loop

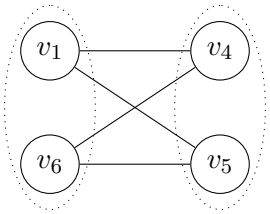


Figure 2.4: Biclique $(\{v_1, v_6\}; \{v_4, v_5\})$.

edge (e.g. $(W_1 = \{v_1\}; W_2 = \{v_4\})$). It contains 4 maximal bicliques of three vertices that can be derived from its maximal cliques $(\{v_1, v_4, v_5\}; \{v_1, v_4, v_5\})$, $(\{v_2, v_4, v_6\}; \{v_2, v_4, v_6\})$, $(\{v_3, v_5, v_6\}; \{v_3, v_5, v_6\})$, and $(\{v_4, v_5, v_6\}; \{v_4, v_5, v_6\})$. It also contains 3 maximal bicliques composed of four vertices $(\{v_1, v_6\}; \{v_4, v_5\})$, $(\{v_2, v_5\}; \{v_4, v_6\})$, $(\{v_3, v_4\}; \{v_5, v_6\})$ and 3 maximal bicliques composed of five vertices $(\{v_4\}; \{v_1, v_2, v_4, v_5, v_6\})$, $(\{v_5\}; \{v_1, v_3, v_4, v_5, v_6\})$, $(\{v_6\}; \{v_2, v_3, v_4, v_5, v_6\})$. Figure 2.4 depicts the subgraph of G_{NO} corresponding to biclique $(\{v_1, v_6\}; \{v_4, v_5\})$.

Remark 4: Isolated cliques are always maximal as they cannot be extended to larger cliques.

Remark 5: Using maximal cliques instead of non-maximal cliques generally leads to a tighter LP relaxation. Constraints based on cliques will therefore be applied to maximal cliques only. Appendix A shows how applying strengthened constraints to maximal cliques only generates the tightest and most compact model. The same remark applies to bicliques.

Remark 6: The number of maximal cliques in an undirected graph might be exponential in the size of the graph. Nevertheless, there are exponential-time algorithms to enumerate all maximal cliques of a graph and we assume the non-overlapping graph is small enough so that this task can be performed in reasonable time. The same remark applies to bicliques.

2.5.2 MOS Model

Aggregated Assignment Constraint Let $W' \subset W$ be a clique of the non-overlapping graph G_{NO}. Then, at most one operation from W' can be assigned to priority-slot i. Therefore, the following constraint, which is at least as strong as constraint (2.4) as shown

in appendix A, is valid.

$$\sum_{v \in W'} Z_{iv} \leq 1 \qquad i \in T, W' \in clique(G_{NO}) \tag{2.17}$$

Aggregated Non-overlapping Constraint Given a clique $W' \subset W$ of the non-overlapping graph G_{NO}, the following aggregated non-overlapping constraint, which is at least as strong as constraint (2.6) as shown in appendix A, is valid. Note that only one term in each summation can be non-zero due to the aggregated assignment constraint (2.17).

$$\sum_{v \in W'} E_{i_1 v} \leq \sum_{v \in W'} S_{i_2 v} + H \cdot (1 - \sum_{v \in W'} Z_{i_2 v}) \quad i_1, i_2 \in T, i_1 < i_2, W' \in clique(G_{NO}) \tag{2.18}$$

Aggregated Non-overlapping Constraint with Intermediate Operations Given a clique $W' \subset W$ of the non-overlapping graph G_{NO}, and two priority-slots $i_1, i_2 \in T$ such that $i_1 < i_2$, constraint (2.18) can be further strengthened (see appendix A) by including the duration of operations assigned to intermediate priority-slots i ($i_1 < i < i_2$).

$$\begin{aligned} &\sum_{v \in W'} E_{i_1 v} + \sum_{\substack{i \in T \\ i_1 < i < i_2}} \sum_{v \in W'} D_{iv} \\ &\leq \sum_{v \in W'} S_{i_2 v} + H \cdot (1 - \sum_{v \in W'} Z_{i_2 v}) \end{aligned} \qquad i_1, i_2 \in T, i_1 < i_2, W' \in clique(G_{NO}) \tag{2.19}$$

Aggregated Non-overlapping Constraint with Clique Transition Times Given a clique $W' \subset W$ of the non-overlapping graph G_{NO}, the clique transition time $TR_{W'}$ is the minimum time delay between any two executions of operations in W'. If W' represents a set of operations executed in a unit, $TR_{W'}$ corresponds to a unit transition time. Clique transition times can be enforced as follows. Note that the value of the big-M constant is

increased to a new valid upper bound for the left hand side of the inequality: $H + TR_{W'}$.

$$\begin{aligned}
&\sum_{v \in W'} (E_{i_1 v} + TR_{W'} \cdot Z_{i_1 v}) \\
&+ \sum_{\substack{i \in T \\ i_1 < i < i_2}} \sum_{v \in W'} (D_{iv} + TR_{W'} \cdot Z_{iv}) \qquad i_1, i_2 \in T, i_1 < i_2, W' \in clique(G_{NO}) \quad (2.20) \\
&\leq \sum_{v \in W'} S_{i_2 v} + (H + TR_{W'}) \cdot (1 - \sum_{v \in W'} Z_{i_2 v})
\end{aligned}$$

2.5.3 MOS-SST Model

Aggregated Synchronization Constraints Given a clique $W' \subset W$ of the non-overlapping graph G_{NO}, the following aggregated synchronization constraints, which are tighter than constraints (2.13) as shown in appendix A, are valid. Note that in each summation only one term can be non-zero.

$$\sum_{v \in W'} S_{iv} \leq t_i \qquad i \in T, W' \in clique(G_{NO}) \qquad (2.21a)$$

$$\sum_{v \in W'} S_{iv} \geq t_i - H \cdot (1 - \sum_{v \in W'} Z_{iv}) \qquad i \in T, W' \in clique(G_{NO}) \qquad (2.21b)$$

Tightening Precedence Constraint Consider a precedence constraint between sets of operations W_1 and W_2 ($P_{W_1 W_2} = 1$). Assume that an operation from W_1 is assigned to priority-slot i_1 with associated time-point t_1 and that an operation from W_2 is assigned to priority-slot i_2 with associated time-point t_2. Then, due to the precedence constraint, we must have $t_1 < t_2$. Therefore, a necessary condition for the precedence constraint to hold true is $i_1 < i_2$. The following constraint ensures that this condition is satisfied, and therefore complements constraint (2.10).

$$\sum_{\substack{j \in T \\ j < i}} \sum_{v \in W_1} Z_{jv} \geq \sum_{\substack{j \in T \\ j \leq i}} \sum_{v \in W_2} Z_{jv} \qquad i \in T, W_1, W_2 \subset W, P_{W_1 W_2} = 1 \qquad (2.22)$$

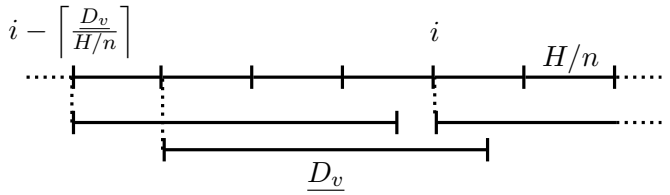


Figure 2.5: Assignment constraint using consecutive time-points.

2.5.4 MOS-FST Model

The use of graph theory to strengthen discrete-time scheduling formulations has been studied by Waterer et al. (2002) who derived facet defining inequalities based on cliques and 5-holes in a finite graph. This graph can be seen as an extension of the non-overlapping graph where each node corresponds to an operation and a time interval (with integer bounds). In this work, we use a basic non-overlapping graph (with no time intervals) which only generates clique-based constraints because it is not possible to enumerate time intervals in continuous-time representations. Therefore, the exact same graph is used for all time representations.

Assignment Constraint using Time-points Consider an operation $v \in W$. If the time interval between two time-points is smaller than the duration of v, then v cannot start at both time-points (v cannot be assigned to both priority-slots). As illustrated in Figure 2.5, v cannot be assigned to priority-slots $i - \left\lceil \frac{D_v}{H/n} \right\rceil + 1$ and i simultaneously, as well as any other priority-slot in between. However, it can be assigned to priority-slots $i - \left\lceil \frac{D_v}{H/n} \right\rceil$ and i simultaneously. Note that the lower bound on the duration of v is used to account for the case of variable processing times. Therefore, the following constraint, which is a stronger extension of constraint (2.4), is valid.

$$Z_{iv} + \sum_{\substack{j \in T \\ i - \delta_v < j < i}} Z_{jv} \leq 1 \qquad\qquad i \in T, v \in W, \delta_v = \left\lceil \frac{D_v}{H/n} \right\rceil \tag{2.23}$$

Aggregated Assignment Constraint The previous constraint (2.23) can be extended to cliques of G_{NO}. For fixed processing times, this constraint is equivalent to equation (9) from Shah et al. (1993).

$$\sum_{v \in W'} \left\{ Z_{iv} + \sum_{\substack{j \in T \\ i-\delta_v < j < i}} Z_{jv} \right\} \leq 1 \qquad i \in T, W' \in clique(W), \delta_v = \left\lceil \frac{D_v}{H/n} \right\rceil \tag{2.24}$$

Aggregated Assignment Constraint with Transitions Times The previous constraint (2.24) can also be extended to the case of clique transition times.

$$\sum_{v \in W'} \left\{ Z_{iv} + \sum_{\substack{j \in T \\ i-\delta_{W'}(v) < j < i}} Z_{jv} \right\} \leq 1 \quad i \in T, W' \in clique(W), \delta_{W'}(v) = \left\lceil \frac{D_v + TR_{W'}}{H/n} \right\rceil \tag{2.25}$$

2.5.5 SOS Model

Aggregated Non-overlapping Constraint Additional aggregated non-overlapping constraints can be generated based on bicliques of the non-overlapping graph for the SOS model. Given a biclique $(W_1; W_2)$ of the non-overlapping graph G_{NO}, the following aggregated non-overlapping constraints, which are at least as strong as constraints (2.5) as shown in appendix A, are valid. Note that in each summation only one term can be non-zero due to the SOS specific assignment constraint (2.16). Also, note that constraints (2.18) and (2.26) are not redundant as they are applied to different sets of operations.

$$\sum_{v \in W_1} E_{i_1 v} \leq \sum_{v \in W_2} S_{i_2 v} + H \cdot (1 - \sum_{v \in W_2} Z_{i_2 v}) \qquad \begin{array}{r} i_1, i_2 \in T, i_1 < i_2, \\ (W_1; W_2) \in biclique(G_{NO}) \end{array} \tag{2.26a}$$

$$\sum_{v \in W_2} E_{i_1 v} \leq \sum_{v \in W_1} S_{i_2 v} + H \cdot (1 - \sum_{v \in W_1} Z_{i_2 v}) \qquad \begin{array}{r} i_1, i_2 \in T, i_1 < i_2, \\ (W_1; W_2) \in biclique(G_{NO}) \end{array} \tag{2.26b}$$

2.6 Solution Methods

Determining the minimum number of priority-slots needed to find the optimal schedule is non-trivial. A commonly used algorithm is to solve several scheduling models, each time increasing the number of priority-slots. We present two different approaches based on this idea followed by the more classic direct approach.

2.6.1 Additive Approach

The MOS, MOS-SST and SOS time representations are such that any solution found with n priority-slots can be found with $n+1$ priority-slots: $z \in \mathbf{MOS}(n) \Rightarrow z \in \mathbf{MOS}(n+1)$. Therefore, the corresponding models can be solved by successively increasing by 1 the number of priority-slot. The additive approach is described by Algorithm 1. The parameter n_0 is the initial number of priority-slots. To improve efficiency of the branch & bound algorithm at each iteration, the cutoff parameter is set using the objective value of the best incumbent found so far. During some iterations, the MILP model may not have any solution strictly better than the best incumbent. In such cases, it will not return any solution even though the model may be feasible.

Three stopping criteria are considered. The first one is $(\Delta \leq \varepsilon)$ where ε is an absolute tolerance on the variation of objective value. In general, this criterion does not guarantee global optimality of the solution even when setting ε to 0. However, it leads to a small number of iterations. Also, in most of our experiments, this stopping criterion returned the global optimal solution, which was only proved optimal by using the following stopping criterion.

The second stopping criterion is $(n > \overline{n})$ where $\overline{n}$ is an upper limit on the number of priority-slots. In some cases, it is possible to determine an upper limit on the number of priority-slots needed to find the optimal solution of the scheduling problem. In such cases, this stopping criterion guarantees global optimality of the solution.

The third stopping criterion is a time limit on the total computational time, which of

Algorithm 1: Additive approach.

```
begin
    z* ⟵ ∅ ;
    cutoff ⟵ −∞ ;
    n ⟵ n_0 ;
    repeat
        z ⟵ Maximize(MOS(n), cutoff) ;
        Δ ⟵ z.objval() − z*.objval() ;
        if Δ > 0 then
            z* ⟵ z ;
            cutoff ⟵ z*.objval() ;
        n ⟵ n + 1 ;
    until stopping condition ;
    return z* ;
end
```

course does not guarantee global optimality.

It is important to note that at each iteration, the integer feasible space of **MOS**(n) includes scheduling solutions explored during previous iterations. In order to avoid redundant search, we introduce constraint (2.27) that rejects any solution that do not make use of all priority-slots. By adding this constraint to the **MOS**(n) model, the property $z \in \textbf{MOS}(n) \Rightarrow z \in \textbf{MOS}(n+1)$ is no longer valid. The same remark holds true for the **MOS-SST**(n) and **SOS**(n) models.

$$\sum_{v \in W} Z_{iv} \geq 1 \qquad\qquad i \in T \qquad\qquad (2.27)$$

2.6.2 Multiplicative Approach

The previous approach could be used to solve a scheduling problem using the MOS-FST time representation. However, a major flaw is that it is not guaranteed that a solution found with n priority-slots can be found with $n+1$ priority-slots. This can be overcome by multiplying the number of priority-slots be a factor of 2 instead of using an increment. Indeed, any solution found with n priority-slots can be found with $2n$ priority-slots: $z \in \textbf{MOS-FST}(n) \Rightarrow z \in \textbf{MOS-FST}(2n)$. The multiplicative approach is therefore very

Algorithm 2: Multiplicative approach.

```
begin
    z* ⟵ ∅ ;
    cutoff ⟵ −∞ ;
    n ⟵ n_0 ;
    repeat
        z ⟵ Maximize(MOS(n), cutoff) ;
        Δ ⟵ z.objval() − z*.objval() ;
        if Δ > 0 then
            z* ⟵ z ;
            cutoff ⟵ z*.objval() ;
        n ⟵ 2 · n ;
    until stopping condition ;
    return z* ;
end
```

similar to the additive approach and is described by Algorithm 2. The stopping criteria introduced for the additive approach can still be used for the multiplicative approach.

2.6.3 Direct Approach

Although the additive approach can be used to solve SOS models, it might not be very efficient. Indeed, each time the number of priority-slot is increased by 1, the solver can only schedule one additional operation. In MOS or MOS-SST models, several additional operations can be scheduled at each iteration, which leaves much more flexibility to better improve the objective value. Instead, a direct approach can be used. It consists of choosing a fixed value for n and solving the MILP model once. In some cases, the total number of executions of operations is fixed and known in advanced, so it can be used as the number of priority-slots, guaranteeing global optimality of the solution obtained. It other cases, a detailed analysis of the scheduling structure of the problem needs to be performed to effectively define a value for n. Global optimality may not be guaranteed if n is too small, or if additional constraints are used to improve the search.

2.7 Single-Stage Batch Scheduling Problem

In this section, we apply the various time representations to model single-stage batch scheduling problems. We study several instances introduced in Pinto and Grossmann (1995). The largest instance has 29 orders to be processed before given due dates from time 0 to time 30. Four units are available to process each single-stage order. Each unit can process only one order at a time and a minimum unit-specific set-up time is required between any two orders. Each order can only be processed on a subset of all units with order-and-unit-specific processing times. Table 2.3 displays all required data. For each order, units that do not have a corresponding processing time cannot be selected for this order. The objective is to minimize total earliness which corresponds to maximizing the end times of all orders. Five instances have been studied with 8, 12, 18, 25, and 29 orders, which we denote SSBSP8, ..., SSBSP29. We introduce the following specific sets.

- $O \subset \{o_1, \ldots, o_{29}\}$ is the set of orders ($O = \{o_1, \ldots, o_8\}$ for SSBSP8).
- $U = \{u_1, \ldots, u_4\}$ is the set of units.
- U_o is the set of units on which order $o \in O$ can be processed. For instance, $U_{o_1} = \{u_1, u_4\}$.
- O_u is the set of orders that can be processed on unit $u \in U$. For instance, $O_{u_3} = \{o_4, o_5, o_7, o_8\}$.

The set of operations W can then be defined as follows. Exactly one operation is defined for each order and each unit on which the order can be processed. This reduces the number of indices from 2 to 1, although the combinatorial size of the problem remains identical.

$$W = \{(o, u) \in O \times U, u \in U_o\} = \{o_1u_1, o_1u_4, o_2u_1, o_2u_4, o_3u_1, o_3u_4, o_4u_3, o_4u_4, \ldots\}$$

Figure 2.6 depicts the non-overlapping graph of SSBSP8. It contains 3 isolated cliques each corresponding to one unit (unit u_2 cannot be used in this instance as it is excluded for all orders, as seen in Table 2.3). The unit-based structure of the non-overlapping isolated cliques leads to classical strengthened assignment and scheduling constraints that have

Table 2.3: Data for single-stage batch scheduling problems.

Order	Due date (hr)	Unit processing time (hr)			
		1	2	3	4
1	15	1.538			1.194
2	30	1.500			0.789
3	22	1.607			0.818
4	25			1.564	2.143
5	20			0.736	1.017
6	30	5.263			3.200
7	21	4.865		3.025	3.214
8	26			1.500	1.440
9	30			1.869	2.459
10	29		1.282		
11	30		3.750		3.000
12	21		6.796	7.000	5.600
13	30	11.250			6.716
14	25	2.632			1.527
15	24	5.000			2.985
16	30	1.250			0.783
17	30	4.474			3.036
18	30		1.429		
19	13		3.130		2.687
20	19	2.424		1.074	1.600
21	30	7.317		3.614	
22	20			0.864	
23	12			3.624	
24	30			2.667	4.000
25	17	5.952		3.448	4.902
26	20	3.824			1.757
27	11	6.410			3.937
28	30	5.500			3.235
29	25				4.286
Set-up times		0.180	0.175	0.000	0.237

already been presented in the literature. Therefore, the mathematical models for batch scheduling problems do not differ very much from previous works. We will denote $W_u = \{v = (o, u), o \in O_u\}$, where $u \in U$, the set of operations executed on unit $u \in U$, W_u is an isolated clique of G_{NO}. Given an order $o \in O$, exactly one execution of this order must be performed. Therefore, the set $W_o = \{v = (o, u), u \in U_o\}$ has an associated cardinality

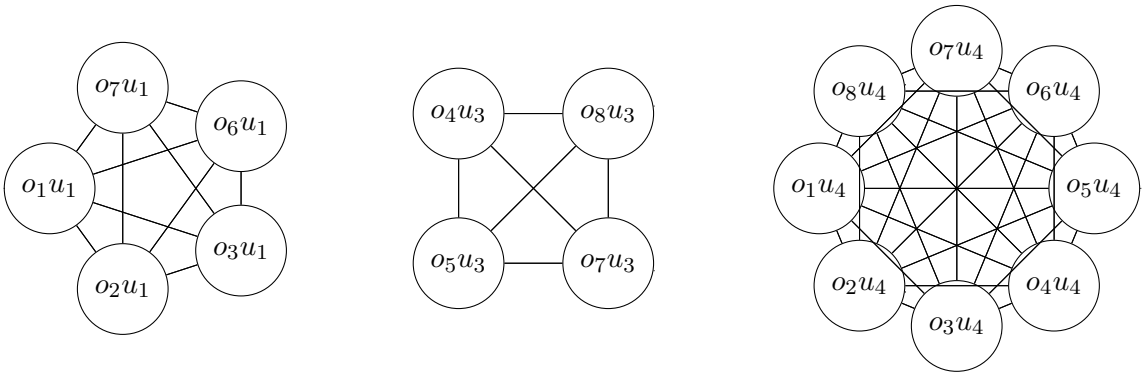


Figure 2.6: Non-overlapping graph with isolated cliques for SSBSP8.

constraint of 1, that is $\underline{N_{W_o}} = \overline{N_{W_o}} = 1$.

2.7.1 MOS Model

The MOS model for the single-stage batch scheduling problem is derived from constraints (2.1)-(2.3), (2.17), (2.20), and (2.27).

$$
\begin{aligned}
\max \quad & \sum_{i\in T}\sum_{v\in W} E_{iv} & & \\
\text{s.t.} \quad & D_{iv} = D_v \cdot Z_{iv} & & i\in T, v\in W \\
& E_{iv} \le \overline{E_v} \cdot Z_{iv} & & i\in T, v\in W \\
& E_{iv} = S_{iv} + D_{iv} & & i\in T, v\in W \\
& \sum_{i\in T}\sum_{v\in W_o} Z_{iv} = 1 & & o\in O \\
& \sum_{v\in W_u} Z_{iv} \le 1 & & i\in T, u\in U \\
& \sum_{v\in W_u} (E_{i_1 v} + TR_{W_u}\cdot Z_{i_1 v}) & & \\
& + \sum_{\substack{i\in T \\ i_1<i<i_2}} \sum_{v\in W_u} (D_{iv} + TR_{W_u}\cdot Z_{iv}) & & i_1, i_2 \in T, i_1 < i_2, u\in U \\
& \le \sum_{v\in W_u} S_{i_2 v} + (H + TR_{W_u})\cdot (1 - \sum_{v\in W_u} Z_{i_2 v}) & & \\
& \sum_{v\in W} Z_{iv} \ge 1 & & i\in T \\
& S_{iv}, D_{iv}, E_{iv} \ge 0 & & i\in T, v\in W \\
& Z_{iv} \in \{0,1\} & & i\in T, v\in W
\end{aligned}
$$

This model can be strengthened in different ways. For each unit, it is possible to derive the minimum and maximum number of orders that will be processed on it. The maximum number of orders that can be processed on unit $u \in U$ is defined as the number of operations in the set.

$$\overline{N_{W_u}} = \left|W_u\right| \qquad u\in U$$

An improved upper cardinality value for W_u can be obtained by considering the processing times of operations that might be processed on unit u. Indeed, due to a limited scheduling horizon, it might not be possible to execute all operations in W_u. Therefore, we use the following improved definition that solves a one-dimensional knapsack problem where all operations have a value of 1. A linear algorithm to solve this problem consists of ordering the operations in W_u with respect to their duration D_v and, starting from the operation

Table 2.4: Unit cardinality bounds depending on parameter n for SSBSP29.

n	u_1	u_2	u_3	u_4
8	[8,8]	[5,5]	[8,8]	[8,8]
9	[6,9]	[2,5]	[6,9]	[6,9]
10	[4,9]	[2,5]	[5,10]	[5,10]
11	[2,9]	[2,5]	[4,11]	[4,11]
12	[1,9]	[2,5]	[3,11]	[4,12]
13	[0,9]	[2,5]	[2,11]	[4,13]
14	[0,9]	[2,5]	[2,11]	[4,14]

with the lowest processing time, setting γ_v to 1 until the knapsack limit $H + TR_{W_u}$ is reached. For the remaining operations, γ_v is set to 0.

$$\overline{N_{W_u}} = \max\left\{ \sum_{v \in W_u} \gamma_v \,\middle|\, \gamma_v \in \{0,1\}, \sum_{v \in W_u} (D_v + TR_{W_u}) \cdot \gamma_v \leq H + TR_{W_u} \right\} \qquad u \in U$$

We denote $W_u^1 = \{v = (o,u) \in W, U_o = \{u\}\}$ the set of operations that can only be processed on unit $u \in U$. The minimum number of orders that will be processed on unit $u \in U$ can be defined as follows.

$$\underline{N_{W_u}} = \left|W_u^1\right| \qquad u \in U$$

However, as the number of orders executed on units different than $u \in U$ is limited, $\underline{N_{W_u}}$ can be increased as follows.

$$\underline{N_{W_u}} = \max\left\{ \left|W_u^1\right|, \left|O\right| - \sum_{\substack{u' \in U \\ u' \neq u}} \min\{n, \overline{N_{W_{u'}}}\} \right\} \qquad u \in U$$

The values obtained for $\underline{N_{W_u}}$ and $\overline{N_{W_u}}$ in problem SSBSP29 are displayed in Table 2.4 as an example. Using $\underline{N_{W_u}}$ and $\overline{N_{W_u}}$, the following cardinality constraint can be derived.

$$\underline{N_{W_u}} \leq \sum_{i \in T} \sum_{v \in W_u} Z_{iv} \leq \overline{N_{W_u}} \qquad u \in U \tag{2.28}$$

Given a unit $u \in U$, W_u is an isolated clique of G_{NO}, so the optimal sequence of operations from W_u is not affected by operations from $W \setminus W_u$. It is therefore possible to reduce

the set of possible sequences of operations from W_u by assigning these operations to the first priority-slots only. In other words, an operation W_u can be assigned to priority-slot i only if an operation from W_u is assigned to priority-slot $i-1$. This leads to the following symmetry-breaking constraint.

$$\sum_{v \in W_u} Z_{iv} \leq \sum_{v \in W_u} Z_{(i-1)v} \qquad i \in T, i \neq 1, u \in U \tag{2.29}$$

This idea can be further applied to other constraints in the model. A maximum cardinality constraint on the set of operations W_u can be enforced by setting to 0 assignment variables corresponding to the last priority-slots for this set.

$$\sum_{v \in W_u} Z_{iv} = 0 \qquad i \in T, i > \overline{N_{W_u}}, u \in U \tag{2.30}$$

Also, a minimum cardinality constraint on W_u can be enforced by assigning exactly one operation to the first priority-slots.

$$\sum_{v \in W_u} Z_{iv} = 1 \qquad i \in T, i \leq \underline{N_{W_u}}, u \in U \tag{2.31}$$

Besides, non-overlapping constraint (2.20) can be tightened for the first priority-slots as follows.

$$\begin{aligned} &\sum_{v \in W_u} (E_{i_1 v} + TR_{W_u} \cdot Z_{i_1 v}) \\ &+ \sum_{\substack{i \in T \\ i_1 < i < i_2}} \sum_{v \in W_u} (D_{iv} + TR_{W_u} \cdot Z_{iv}) \qquad i_1, i_2 \in T, i_1 < i_2 \leq \underline{N_{W_u}}, u \in U \\ &\leq \sum_{v \in W_u} S_{i_2 v} \end{aligned} \tag{2.32}$$

The single-stage batch scheduling problems are solved with the MOS model plus constraints (2.28)-(2.32). The additive approach is used with the second stopping criterion ($n > \max_u \overline{N_u}$) which guarantees global optimality of the solution. Indeed, for each unit $u \in U$, no more than $\overline{N_u}$ priority-slots are needed to sequence operations on it. The initial number of priority-slots is set to $n_0 = \lceil |O| / |\{u | W_u \neq \emptyset\}| \rceil$ which is the minimum number

of orders at least one unit has to process. Computational results are given in Table 2.5. Experiments were run on an Intel Xeon 1.86GHz processor using GAMS/CPLEX 11. For each iteration, the LP relaxation, MILP solution, number of nodes, CPU time and cumulative CPU time are displayed. The term "no solution" for MILP solution is used when CPLEX did not find any solution with higher objective value than the cutoff value which does not mean that no solution exists. For each problem, the global optimal solution is underlined. The CPU time elapsed until the first stopping criterion ($\Delta \leq 0$) is satisfied is also underlined. The results show that the first stopping criterion leads to significant reduction of CPU times compared to the second. Also, it is interesting to note the problem with 29 orders, although larger, is solved faster than the 25 orders problem. As the scheduling horizon is fixed to 30 hours, the density of operations is higher with 29 orders. Therefore, this problem is more constrained and has potentially fewer solutions. The model size is increased but the branch & bound tree is smaller, thus leading to smaller CPU times. This is verified by the fact that fewer nodes are explored.

Additionally, we solved problem SSBSP18 using the full MOS models with or without the minimum priority-slot usage constraint (2.27). The results obtained with the additive approach are displayed in Figure 2.7. It shows that constraint (2.27) greatly reduces the search space at each iteration. In particular, using this constraint, the last iterations are solved in few seconds, whereas CPU times are higher than 100 seconds if it is not used.

We also solved the SSBSP18 instance with exactly 5 priority-slots using the full MOS model without and with symmetry-breaking constraints, which corresponds to the first iteration of the additive algorithm for this problem. Without symmetry-breaking constraints, the problem was solved in 115 seconds (124788 nodes) as opposed to less than 2 seconds (513 nodes) if symmetry-breaking constraints (2.29)-(2.32) are used. This shows why using symmetry-breaking concepts is crucial to solve scheduling problems as they tend to be highly degenerate (Kallrath, 2002).

We performed another experiment which consisted of solving the full MOS model by using the original assignment constraint (2.4) instead of the strengthened assignment constraint

Table 2.5: MOS computational results for single-stage batch scheduling problems.

Pb	n	LP	MILP	Nb of nodes	CPU	Cumulative CPU
SSBSP8	3	189.000	189.000	3	0.81s	0.81s
	4	189.000	no solution	0	0.77s	1.58
	5	189.000	no solution	0	0.80s	2.38s
	6	189.000	no solution	0	0.76s	3.14s
	7	189.000	no solution	0	0.86s	4.00s
	8	188.035	no solution	0	0.90s	4.90s
SSBSP12	3	298.517	296.543	0	0.83s	0.83s
	4	299.000	297.974	604	1.67s	2.50s
	5	299.000	no solution	207	1.38s	3.88s
	6	299.000	no solution	198	1.71s	5.59s
	7	299.000	no solution	0	1.17s	6.76s
	8	299.000	no solution	0	1.02s	7.78s
	9	no solution			0.95s	8.73s
	10	no solution			1.01s	9.74s
	11	no solution			1.09s	10.83s
SSBSP18	5	461.563	450.760	513	1.83s	1.83s
	6	463.993	450.982	1952	5.71s	7.54s
	7	467.196	451.504	2924	8.37s	15.91s
	8	467.966	no solution	4600	16.84s	32.75s
	9	467.966	no solution	4600	18.47s	51.22
	10	459.877	no solution	0	1.35s	52.57
	11	no solution			1.32s	53.89s
	12	no solution			1.48s	55.37s
	13	no solution			1.72s	57.09s
	14	no solution			1.90s	58.99s
SSBSP25	7	593.615	575.929	3444	17.24s	17.24s
	8	601.884	579.089	18261	69.90s	87.14s
	9	607.713	579.570	55046	189.66s	276.80s
	10	609.000	no solution	77800	408.97s	685.77s
	11	609.000	no solution	22400	113.60s	799.37s
	12	597.717	no solution	0	3.61s	802.98s
	13	583.289	no solution	0	2.31s	805.29s
	14	no solution			2.54s	807.83s
SSBSP29	8	649.531	628.413	1583	12.97s	12.97s
	9	656.403	634.964	10870	56.16s	69.13s
	10	662.720	635.104	42249	206.66s	275.79s
	11	666.512	no solution	22400	110.83s	386.62s
	12	666.552	no solution	3700	32.72s	419.34s
	13	668.164	no solution	1800	28.33s	447.67s
	14	653.051	no solution	0	4.72s	452.39s

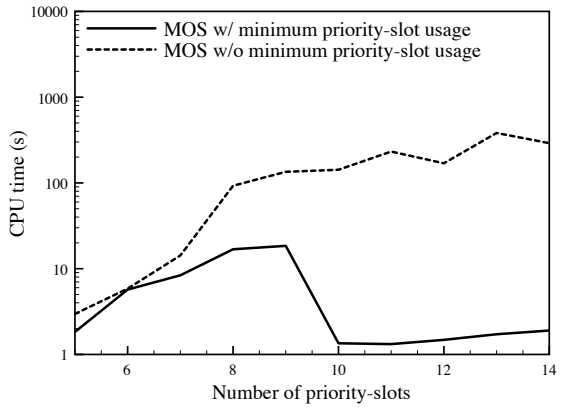


Figure 2.7: Effect of the minimum priority-slot usage constraint.

(2.17). Instance SSBSP18 was solved in 3.43 seconds (587 nodes) instead of 1.83 seconds (513 nodes) for the strengthened constraint. Thus, this constraint slightly helps to improve the solution time although CPLEX is able to generate cuts to tighten the LP relaxation accordingly. However, we also solved the full MOS model replacing the strengthened non-overlapping constraints (2.20) and (2.32) by the following non-overlapping constraint with unit-dependent transition time, which is based on constraint (2.8).

$$\begin{aligned} &E_{i_1v_1} + E_{i_1v_2} + TR_{W_u} \cdot (Z_{i_1v_1} + Z_{i_1v_2}) \\ &\leq S_{i_2v_1} + S_{i_2v_2} + (H + TR_{W_u}) \cdot (1 - Z_{i_2v_1} - Z_{i_2v_2}) \end{aligned} \qquad i_1, i_2 \in T, i_1 < i_2, u \in U, v_1, v_2 \in W_u \tag{2.33}$$

The model was solved in 3,288 seconds and 1,078,034 nodes (vs 1.83 seconds and 513 nodes) which proves that CPLEX was not successful at improving the LP relaxation of the model for this mixed-integer constraint. In general, MILP solvers are very efficient at solving IPs as they are able to exploit the structure of the model in order generate very tight cuts (such as clique constraints, Nemhauser and Wolsey, 1999) that can lead to large improvements of the branch & bound search. However, it is much harder to generate tightening cuts from constraint (2.33) using the clique structure. Therefore, using strengthened formulations for mixed-integer constraints is very important as it greatly improves the performance of MILP solvers.

2.7.2 MOS-SST Model

The MOS-SST model for the single-stage batch scheduling problem is derived from constraints (2.1)-(2.3), (2.12), (2.17), (2.20), (2.21), and (2.27).

$$
\begin{aligned}
\max \quad & \sum_{i\in T}\sum_{v\in W} E_{iv} && \\
\text{s.t.} \quad & D_{iv} = D_v \cdot Z_{iv} && i\in T, v\in W \\
& E_{iv} \le \overline{E_v} \cdot Z_{iv} && i\in T, v\in W \\
& E_{iv} = S_{iv} + D_{iv} && i\in T, v\in W \\
& \sum_{i\in T}\sum_{v\in W_o} Z_{iv} = 1 && o\in O \\
& \underline{N_u} \le \sum_{i\in T}\sum_{v\in W_u} Z_{iv} \le \overline{N_u} && u\in U \\
& \sum_{v\in W_u} Z_{iv} \le 1 && i\in T, u\in U \\
& \sum_{v\in W_u} (E_{i_1 v} + TR_{W_u}\cdot Z_{i_1 v}) & & \\
& + \sum_{\substack{i\in T \\ i_1<i<i_2}} \sum_{v\in W_u} (D_{iv} + TR_{W_u}\cdot Z_{iv}) && i_1, i_2 \in T, i_1 < i_2, u\in U \\
& \le \sum_{v\in W_u} S_{i_2 v} + (H + TR_{W_u})\cdot(1 - \sum_{v\in W_u} Z_{i_2 v}) & & \\
& t_{i-1} \le t_i && i\in T, i\neq 1 \\
& \sum_{v\in W_u} S_{iv} \le t_i && i\in T, u\in U \\
& \sum_{v\in W_u} S_{iv} \ge t_i - H\cdot(1 - \sum_{v\in W'} Z_{iv}) && i\in T, u\in U \\
& \sum_{v\in W} Z_{iv} \ge 1 && i\in T \\
& S_{iv}, D_{iv}, E_{iv} \ge 0 && i\in T, v\in W \\
& Z_{iv} \in \{0,1\} && i\in T, v\in W \\
& t_i \in [0, H] && i\in T
\end{aligned}
$$

Due to the use of time-points, the symmetry-breaking constraints developed in section 2.7.1 cannot be applied to the MOS-SST model. Similarly to the MOS model, the additive approach is used and the initial number of priority-slots is set to

Table 2.6: MOS-SST computational results for single-stage batch scheduling problems.

Pb	n	LP	MILP	Nb of nodes	CPU	Cumulative CPU
SSBSP8	3	189.000	177.001	1,220	1.47s	1.47s
	4	189.000	183.149	6,079	3.48s	4.95s
	5	189.000	185.567	16,339	7.44s	12.39s
	6	189.000	187.815	8,127	10.96s	23.35s
	7	189.000	188.823	25,640	18.38s	41.73s
	8	189.000	189.000	649	2.64s	44.37s
SSBSP12	3	297.770	272.902	33	0.86s	0.86s
	4	299.000	288.031	24,452	19.76s	20.62s
	5	299.000	290.640	195,160	145.87s	166.49s
	6	299.000	292.697	507,940	394.78s	561.27s
	7	299.000	294.289	815,078	716.90s	1,278.17s
SSBSP18	5	468.000	428.544	256,375	476.74s	476.74s
	6	468.000	433.583	360,510	+1,000s	+1,476.74s
SSBSP25	7	609.000	538.538	103,912	+1,000s	+1,000s
SSBSP29	8	638.975	569.580	140,638	+1,000s	+1,000s

$n_0 = \lceil |O| / |\{u | W_u \neq \emptyset\}| \rceil$. To solve the problem to global optimality, we use the second criterion ($n > |O|$). However, all instances except the first one are very expensive to solve, so we used a time limit of 1,000 seconds. Computational results are given in Table 2.6. For each problem, the MOS-SST solution obtained with n priority-slots has lower objective value than the MOS solution obtained with the same number of priority-slots as stated previously. Overall results show that the MOS-SST time representation is much less efficient than the MOS time representation.

2.7.3 MOS-FST Model

The MOS-FST model for the single-stage batch scheduling problem is derived from constraints (2.1)-(2.3), (2.15), and (2.25). Note that non-overlapping constraint (2.20) is not included in the model as assignment constraint (2.25) is sufficient to enforce non-overlapping constraints when processing times are constant ($\underline{D_v} = \overline{D_v}$). Also, constraint (2.27) is not included in the model as valid schedules containing time periods with no activity would become infeasible.

$$
\begin{aligned}
\max \quad & \sum_{i\in T}\sum_{v\in W} E_{iv} & & \\
\text{s.t.} \quad & D_{iv} = D_v \cdot Z_{iv} & & i\in T, v\in W \\
& E_{iv} \le \overline{E_v} \cdot Z_{iv} & & i\in T, v\in W \\
& E_{iv} = S_{iv} + D_{iv} & & i\in T, v\in W \\
& \sum_{i\in T}\sum_{v\in W_o} Z_{iv} = 1 & & o\in O \\
& \underline{N_u} \le \sum_{i\in T}\sum_{v\in W_u} Z_{iv} \le \overline{N_u} & & u\in U \\
& S_{iv} = t_i \cdot Z_{iv} & & i\in T, v\in W \\
& \sum_{v\in W_u}\left\{ Z_{iv} + \sum_{\substack{j\in T \\ i-\delta_u(v)<j<i}} Z_{jv} \right\} \le 1 & & i\in T, u\in U, \delta_u(v) = \left\lceil \frac{D_v + TR_u}{H/n} \right\rceil \\
& S_{iv}, D_{iv}, E_{iv} \ge 0 & & i\in T, v\in W \\
& Z_{iv} \in \{0,1\} & & i\in T, v\in W \\
& t_i = \frac{i-1}{n}\cdot H & & i\in T
\end{aligned}
$$

Similarly to the MOS-SST model, it is not possible to develop symmetry-breaking constraints based on the isolated cliques of G_{NO} for this model. To solve it, the multiplicative approach is used and the initial number of priority-slots is set to $n_0 = \frac{H}{2} = 15$. Due to memory limitations we used the second stopping criterion ($n > 1920$) which does not guarantee global optimality of the solution. Computational results are given in Table 2.7. The first iterations are always computationally inexpensive and lead to good feasible solutions, which are improved in the next iterations. Although near-optimal solutions are obtained quickly, the gap is never entirely closed due to the time discretization. It is also interesting to note that most problems are solved at the root node.

2.7.4 SOS Model

The SOS time representation has not been studied in the case for batch scheduling problems as it requires to develop a problem-specific symmetry-breaking sequencing rule (see

Table 2.7: MOS-FST computational results for single-stage batch scheduling problems.

Pb	n	LP	MILP	Nb of nodes	CPU	Cumulative CPU
SSBSP8	15	182.581	182.581	0	0.80s	0.80s
	30	186.013	186.013	0	1.07s	1.87s
	60	187.456	187.456	0	1.56s	3.43s
	120	188.594	188.594	0	2.77s	6.20s
	240	188.719	188.719	0	6.07s	12.27s
	480	188.813	188.813	0	16.89s	29.16s
	960	188.907	188.907	0	52.92s	82.08s
	1,920	188.954	188.954	0	183.32s	265.40s
SSBSP12	15	288.482	288.482	0	1.02s	1.02s
	30	292.671	292.671	0	1.43s	2.45s
	60	295.466	295.466	0	2.05s	4.50s
	120	296.393	296.393	0	3.70s	8.20s
	240	297.268	297.268	0	8.83s	17.03s
	480	297.675	297.675	0	25.44s	42.47s
	960	297.772	297.772	0	80.46s	122.93s
	1,920	297.878	297.878	0	282.79s	405.72s
SSBSP18	15	429.377	429.377	0	1.31s	1.31s
	30	438.507	438.507	0	1.74s	3.05s
	60	445.652	445.182	0	2.67s	5.72s
	120	448.808	448.107	0	5.30s	11.02s
	240	450.308	449.607	0	14.15s	25.17s
	480	451.339	450.782	0	39.90s	65.07s
	960	451.652	451.157	0	138.20s	203.27s
	1,920	451.800	451.297	0	459.25s	662.52s
SSBSP25	15	537.852	536.490	0	1.39s	1.39s
	30	554.266	536.490	0	2.05s	3.44s
	60	566.344	566.085	0	3.36s	6.80s
	120	573.141	573.091	0	7.07s	13.87s
	240	576.435	576.435	0	17.66s	31.53s
	480	578.464	578.373	0	50.89s	82.42s
	960	578.996	578.904	0	168.31s	250.73s
	1,920	579.300	579.185	0	628.46s	879.19s
SSBSP29	15	570.475	567.485	0	1.47s	1.47s
	30	597.973	595.846	0	2.18s	3.65s
	60	616.779	613.610	8	4.33s	7.98s
	120	627.674	625.258	5	10.02s	18.00s
	240	631.863	630.988	0	22.60s	40.60s
	480	634.336	633.426	0	75.74s	116.34s
	960	635.119	634.019	512	389.57s	505.91s
	1,920	635.741	no solution	400	+1,000s	+1,505.91s

Priority-slots	1	2	3
Unit 1	a	b	c
Unit 2	d	e	f
Unit 3	g	h	i
Unit 4	j	k	l

MOS Assignment

Priority-slots	1	2	3	4	5	6	7	8	9	10	11	12
Unit 1	a	b	c									
Unit 2				d	e	f						
Unit 3							g	h	i			
Unit 4										j	k	l

SOS Assignment

Figure 2.8: Equivalent MOS and SOS assignments for SSBSP12.

chapter 4). However, the following comments can be made. If the number of batches to be processed on each unit is known a priori, the problem reduces to a set of independent sequencing problem for each unit. This is also true for in the MOS time representation. In fact, it is possible to break symmetries in the SOS model in order to make it equivalent to the MOS model as shown in Figure 2.8. In the MOS model, the 3 priority-slots are assigned *in parallel* as each priority-slot is assigned to several operations, at most one for each unit. In the SOS model, the 12 priority-slots are assigned *in sequence* as each priority-slot can be assigned to at most one operation (i.e. at most one unit). Consider a sequencing rule for the SOS model that *reserves* priority-slots 1, 2, and 3 to unit 1, priority-slots 4, 5, and 6 to unit 2, and so on. Grey cells in Figure 2.8 are used to represent the assignments that are forbidden by such a sequencing rule. In this case, there exists a bijective transformation between MOS and SOS assignments as outlined by the lower-case letters. Both assignments lead to identical precedence relations between assigned operations. A similar idea can be used when the number of batches to be processed on each unit in not known a priori. It should be noted that there are no non-trivial bicliques in the non-overlapping graph G_{NO}. Furthermore, the same comments apply to the multi-stage batch scheduling problem studied in section 2.8.

2.7.5 Models Comparison

Figure 2.9 shows how the objective value of the best incumbent varies over time. A step increase of the objective value corresponds to a new solution found during the solution algorithm. Note that we do not include intermediate solutions found during each branch &

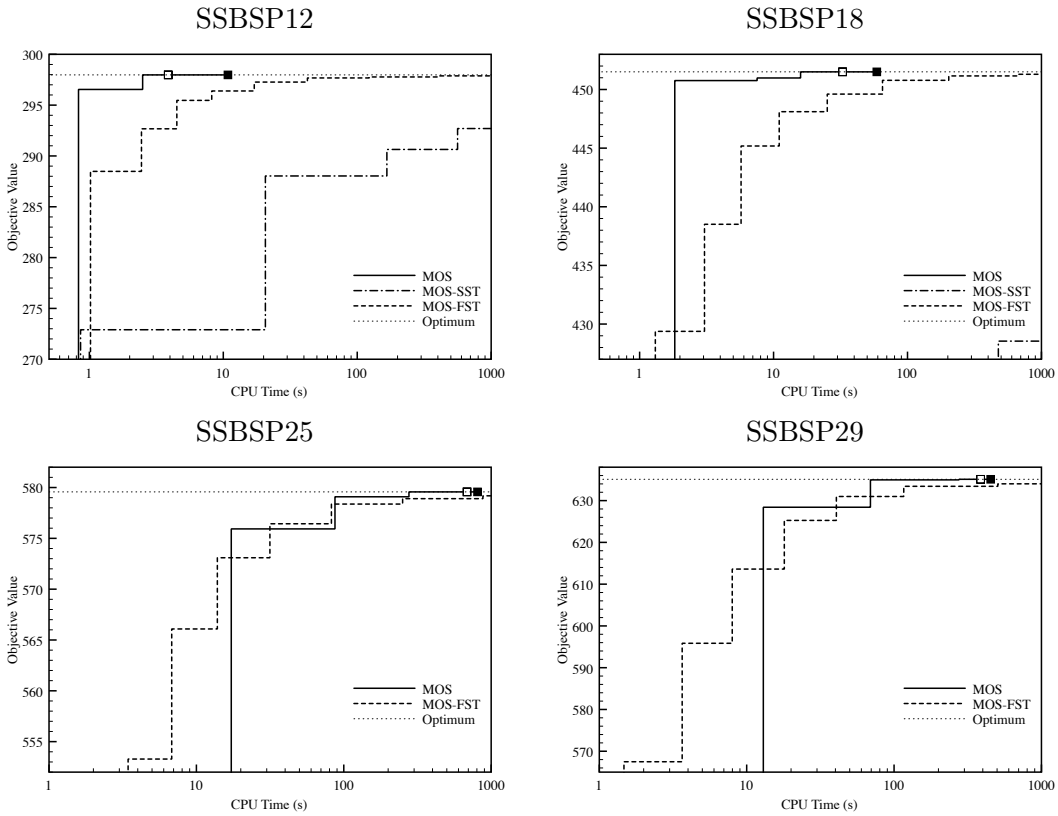


Figure 2.9: Comparison of time representations for single-stage batch scheduling problems.

bound search. Empty squares represent the time when the first stopping criterion is satisfied while plain squares represent the time when the second stopping criterion is satisfied. This applies only to the MOS model. From this figure, it is clear that the MOS model is superior to the other models as it finds first feasible solutions very fast, and is able to find optimal solutions and prove their optimality in reasonable time. The MOS-FST model compares well in particular for finding first feasible solutions. It also finds near-optimal solutions in reasonable time although it was slower than the MOS model for the instances that were considered. Also, it is clear that the MOS-SST representation is not well suited to solve these problems. For a more extended review of the performance of the different models available to solve single-stage batch scheduling problems, the reader may refer to Méndez and Cerdá (2003) and Castro and Grossmann (2006).

2.8 Multi-Stage Batch Scheduling Problem

We now extend the previous study to multi-stage batch scheduling problems from Pinto and Grossmann (1995). The largest instance has 10 orders to be processed sequentially in 5 consecutive stages before identical due dates $t = 500$ hr. Twenty-five units are available to process each order. Each unit is associated to one stage and can process only one order at a time. A minimum unit-specific set-up time is required between any two orders processed on the unit. Each order can only be processed on a subset of all units with order-and-unit-specific processing times. Table 2.8 displays all required data. For each order, units that do not have a corresponding processing time cannot be selected for this order. The objective, as introduced by Pinto and Grossmann (1995), is to minimize total weighted earliness which corresponds to maximizing a weighted summation of the end time of all orders in all stages. The weights are stage dependent and defined as $w_l = 0.2 \cdot l$. Three instances have been studied with 5, 8 and 10 orders, which we denote MSBSP5, MSBSP8 and MSBSP10. We introduce the following specific sets.

- $O \subset \{o_1, \ldots, o_10\}$ is the set of orders ($O = \{o_1, \ldots, o_5\}$ for MSBSP5).
- $L = \{l_1, \ldots, l_5\}$ is the set of stages
- $U = \{u_1, \ldots, u_{25}\}$ is the set of units.
- U_o is the set of units on which order $o \in O$ can be processed. For instance, $U_{o_1} = \{u_1, \ldots, u_{17}, u_{20}, u_{21}, u_{23}, u_{24}\}$.
- O_u is the set of orders that can be processed on unit $u \in U$. For instance, $O_{u_{20}} = \{o_1, o_3, \ldots, o_9\}$.
- l_u is the stage in which unit $u \in U$ can be used. For instance, $l_{u_7} = l_2$
- U_l is the set of units which can be used in stage $l \in L$. For instance, $U_{l_5} = \{u_{23}, u_{24}, u_{25}\}$

The set of operations can then be defined as follows. Exactly one operation is defined for each order, each stage and each unit on which the order can be processed. This reduces the number of indices from 3 to 1, although the combinatorial size of the problem remains

Table 2.8: Data for multi-stage batch scheduling problems.

Stage	Unit	Transition Time (hr)	Order processing time (hr)									
			1	2	3	4	5	6	7	8	9	10
1	1	8	18.1	23	18.1	20	17	15	31	12	13	12
	2	8	18.1	23	18.1	20	17	14	30	12	7	4
	3	8	18.1	23	18.1	20	17	13	34	14	8	23
	4	8	18.1	23	18.1	20	17	12	32	15	9	12
	5	8	18.1	23	18.1	20	17	18	31			
	6	8	18.1	23	18.1	20	17	15		16	16	14
2	7	8	14	14	14	11	14	15	31		15	13
	8	1	5	5	5	5	5	7	31	16	15	13
	9	1	5	5	5	5	5	7	31	15	11	13
3	10	2.5	12	12	24	12	12	13	14	12	13	12
	11	2.5	12	12	24	12	12	12	15	13	7	4
	12	2.5	12	12	24	12	12	15	16	14	8	23
	13	2.5	12	12	24	12	12	17	41	14	9	12
	14	2.5	12	12	24	12	12	17	15	14		
	15	2.5	12	12	24	12	12	18	81	14	16	14
	16	2.5	12	12	24	12	12	19		14	15	13
	17	2.5	12	12	24	12	12		16	14	15	13
	18	2.5		12				16	16	14	11	13
	19	2.5		12				13	21	10	12	6
4	20	6	9.5		9.3	7.9	12.5	13.5	12	10	15	
	21	6	9.5		9.3	7.9	12.5	14	13	9	17	12
	22	24		100				14.5	11	8	17	23
5	23	4	24		24	24	24	12	11		22	12
	24	4	24		24	24	24	12	11	7	21	22
	25	5		48				23	11	7		12

identical.

$$W = \{(o, l, u) \in O \times L \times U, u \in U_o \cap U_l\}$$
$$= \{o_1 l_1 u_1, \ldots, o_1 l_1 u_6, o_1 l_2 u_7, \ldots, o_1 l_2 u_9, o_1 l_3 u_{10}, \ldots\}$$

Figure 2.10 partially depicts the non-overlapping graph of MSBSP5. It contains 25 isolated cliques each corresponding to one unit. We will denote $W_u = \{v = (o, l, u), o \in O_u, l = l_u\}$, where $u \in U$, the set of operations executed on unit u, W_u is an isolated clique of G_{NO}. Given an order $o \in O$ and a stage $l \in L$, exactly one execution of order o in stage l must be

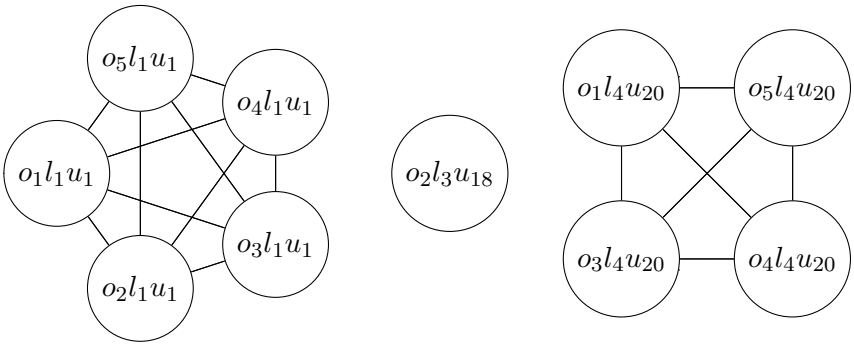


Figure 2.10: Partial non-overlapping graph with isolated cliques for MSBSP5.

performed. Therefore, the set $W_{ol} = \{v = (o, l, u), u \in U_o \cap U_l\}$ has an associated cardinality constraint of 1, that is $\underline{N_{W_{ol}}} = \overline{N_{W_{ol}}} = 1$. Furthermore, due to the multi-stage feature of this problem, precedence constraints exist between operations belonging to different stages of the same order. More precisely, for each order $o \in O$ and each stage $l \in L \setminus \{l_1\}$, sets $W_{o(l-1)}$ and W_{ol} must satisfy a precedence constraint $P_{W_{o(l-1)}W_{ol}} = 1$.

2.8.1 MOS Model

The MOS model for the multi-stage batch scheduling problem is derived from constraints (2.1)-(2.3), (2.10), (2.17), (2.20), (2.27), and (2.29)-(2.32).

$$
\begin{aligned}
\max \quad & \sum_{i \in T} \sum_{\substack{v \in W \\ v=(o,l,u)}} 0.2 \cdot l \cdot E_{iv} & & \\
\text{s.t.} \quad & D_{iv} = D_v \cdot Z_{iv} & & i \in T, v \in W \\
& E_{iv} \leq \overline{E_v} \cdot Z_{iv} & & i \in T, v \in W \\
& E_{iv} = S_{iv} + D_{iv} & & i \in T, v \in W \\
& \sum_{i \in T} \sum_{v \in W_{ol}} Z_{iv} = 1 & & o \in O, l \in L \\
& \underline{N_u} \leq \sum_{i \in T} \sum_{v \in W_u} Z_{iv} \leq \overline{N_u} & & u \in U \\
& \sum_{i \in T} \sum_{v \in W_{o(l-1)}} E_{iv} \leq \sum_{i \in T} \sum_{v \in W_{ol}} S_{iv} & & o \in O, l \in L, l \neq 1 \\
& \sum_{v \in W_u} Z_{iv} \leq 1 & & i \in T, u \in U \\
& \sum_{v \in W_u} (E_{i_1 v} + TR_{W_u} \cdot Z_{i_1 v}) & & \\
& + \sum_{\substack{i \in T \\ i_1 < i < i_2}} \sum_{v \in W_u} (D_{iv} + TR_{W_u} \cdot Z_{iv}) & & i_1, i_2 \in T, i_1 < i_2, i_2 > \underline{N_{W_u}}, u \in U \\
& \leq \sum_{v \in W_u} S_{i_2 v} + (H + TR_{W_u}) \cdot (1 - \sum_{v \in W_u} Z_{i_2 v}) & & \\
& \sum_{v \in W} Z_{iv} \geq 1 & & i \in T \\
& \sum_{v \in W_u} Z_{iv} \leq \sum_{v \in W_u} Z_{(i-1)v} & & i \in T, i \neq 1, u \in U \\
& \sum_{v \in W_u} Z_{iv} = 0 & & i \in T, i > \overline{N_{W_u}}, u \in U \\
& \sum_{v \in W_u} Z_{iv} = 1 & & i \in T, i \leq \underline{N_{W_u}}, u \in U \\
& \sum_{v \in W_u} (E_{i_1 v} + TR_{W_u} \cdot Z_{i_1 v}) & & \\
& + \sum_{\substack{i \in T \\ i_1 < i < i_2}} \sum_{v \in W_u} (D_{iv} + TR_{W_u} \cdot Z_{iv}) & & i_1, i_2 \in T, i_1 < i_2, i_2 \leq \underline{N_{W_u}}, u \in U \\
& \leq \sum_{v \in W_u} S_{i_2 v} & & \\
& S_{iv}, D_{iv}, E_{iv} \geq 0 & & i \in T, v \in W \\
& Z_{iv} \in \{0, 1\} & & i \in T, v \in W
\end{aligned}
$$

Cardinality bounds for units are defined similarly to the single-stage case. The minimum cardinality bound is obtained by considering the number of orders executed on units different

Table 2.9: MOS computational results for multi-stage batch scheduling problems.

Pb	n	LP	MILP	Nb of nodes	CPU	Cumulative CPU
MSBSP5	2	6,827.56	6,827.56	0	0.98s	0.98s
	3	6,828.76	6,828.76	58	1.66s	2.64
	4	6,996.76	no solution	0	2.26s	4.90s
	5	6,996.76	no solution	0	3.23s	8.13s
MSBSP8	3	10,985.61	10,985.16	837	16.70s	16.70s
	4	11,364.67	10,986.36	1,149	59.87s	76.57s
	5	11,364.67	no solution	500	30.34s	106.91s
	6	11,364.67	no solution	2,200	156.14s	263.05s
	7	11,364.67	no solution	1,500	243.65s	506.70s
	8	11,364.67	no solution	2,200	331.18s	837.88s
MSBSP10	4	13,627.41	13,581.16	34,340	+1,000s	+1,000s

than $u \in U$ that belong to the same stage l_u.

$$\overline{N_{W_u}} = \max \left\{ \sum_{v \in W_u} \gamma_v \,\middle|\, \gamma_v \in \{0,1\}, \sum_{v \in W_u} (D_v + TR_{W_u}) \cdot \gamma_v \leq H + TR_{W_u} \right\} \qquad u \in U$$

$$\underline{N_{W_u}} = \max \left\{ \left|W_u^1\right|, \left|O\right| - \sum_{\substack{u' \in U \\ u' \neq u \\ l_{u'} = l_u}} \min\{n, \overline{N_{W_{u'}}}\} \right\} \qquad u \in U$$

The multi-stage batch scheduling problems are solved using the additive approach with the second stopping criterion ($n > \max_u \overline{N_u}$) which guarantees global optimality of the solution. The initial number of priority-slots is set to $n_0 = \left\lceil \left|O\right| / \min_{l \in L} \left|\{u | u \in U_l\}\right| \right\rceil$, which is the minimum number of orders at least one unit has to process in each stage, more precisely in bottleneck stages. Computational results are given in Table 2.9. Similarly to the single-stage case, the results show that the first stopping criterion leads to significant reduction of CPU times compared to the second. The instance with 10 orders cannot be solved to global optimality although feasible solutions are obtained quickly using 4 priority-slots. The branch & bound search is stopped after 1,000 seconds with a remaining 0.09% optimality gap. A 1% optimality gap can be achieved in 219.08 seconds.

2.8.2 MOS-SST Model

The MOS-SST model for the multi-stage batch scheduling problem is derived from constraints (2.1)-(2.3), (2.10), (2.17), (2.20)-(2.22), and (2.27).

$$
\begin{aligned}
\max \quad & \sum_{i\in T}\sum_{\substack{v\in W\\ v=(o,l,u)}} 0.2\cdot l\cdot E_{iv} && \\
\text{s.t.} \quad & D_{iv} = D_v \cdot Z_{iv} && i\in T, v\in W\\
& E_{iv} \le \overline{E_v}\cdot Z_{iv} && i\in T, v\in W\\
& E_{iv} = S_{iv} + D_{iv} && i\in T, v\in W\\
& \sum_{i\in T}\sum_{v\in W_{ol}} Z_{iv} = 1 && o\in O, l\in L\\
& \underline{N_u} \le \sum_{i\in T}\sum_{v\in W_u} Z_{iv} \le \overline{N_u} && u\in U\\
& \sum_{i\in T}\sum_{v\in W_{o(l-1)}} E_{iv} \le \sum_{i\in T}\sum_{v\in W_{ol}} S_{iv} && o\in O, l\in L, l\neq 1\\
& \sum_{v\in W_u} Z_{iv} \le 1 && i\in T, u\in U\\
& \sum_{v\in W_u}(E_{i_1v} + TR_{W_u}\cdot Z_{i_1v}) && \\
& + \sum_{\substack{i\in T\\ i_1<i<i_2}}\sum_{v\in W_u}(D_{iv} + TR_{W_u}\cdot Z_{iv}) && i_1,i_2\in T, i_1<i_2, u\in U\\
& \le \sum_{v\in W_u} S_{i_2v} + (H + TR_{W_u})\cdot(1-\sum_{v\in W_u} Z_{i_2v}) && \\
& t_{i-1} \le t_i && i\in T, i\neq 1\\
& \sum_{v\in W_u} S_{iv} \le t_i && i\in T, u\in U\\
& \sum_{v\in W_u} S_{iv} \ge t_i - H\cdot(1-\sum_{v\in W'} Z_{iv}) && i\in T, u\in U\\
& \sum_{\substack{j\in T\\ j<i}}\sum_{v\in W_{o(l-1)}} Z_{jv} \ge \sum_{\substack{j\in T\\ j\le i}}\sum_{v\in W_{ol}} Z_{jv} && i\in T, o\in O, l\in L, l\neq 1\\
& \sum_{v\in W} Z_{iv} \ge 1 && i\in T\\
& S_{iv}, D_{iv}, E_{iv} \ge 0 && i\in T, v\in W\\
& Z_{iv}\in\{0,1\} && i\in T, v\in W\\
& t_i \in [0,H] && i\in T
\end{aligned}
$$

Table 2.10: MOS-SST computational results for multi-stage batch scheduling problems.

Pb	n	LP	MILP	Nb of nodes	CPU	Cumulative CPU
MSBSP5	5	no solution			1.45s	1.45s
	6	6,617.96	6,161.40	21,915	21.89s	23.34s
	7	6,996.76	6,448.62	285,564	+1,000s	+1,023.34s
MSBSP8	5	no solution			1.81s	1.81s
	6	no solution			2.07s	3.88s
	7	11,363.98	99,79.40	21,491	+1,000s	+1,003.88s
MSBSP10	5	no solution			2.19s	2.19s
	6	no solution			2.37s	4.56s
	7	no solution			3.36s	7.92s
	8	14,255.76	12,560.20	2,172	+1,000s	+1,007.92s

The additive approach is used to solve the model and the initial number of priority-slots is set to $n_0 = |L|$, the number of stages, as for each order stage processing operations must start at different dates. No problem instance was solved to global optimality, so we use a time limit of 1,000 seconds. Computational results are given in Table 2.10. As for the single-stage case, the MOS-SST time representation is much less efficient than the MOS time representation. For the instance with 10 orders the MOS-SST model returns a worse solution than the MOS model in 1,000 seconds (13.31% optimality gap remaining).

2.8.3 MOS-FST Model

The MOS-FST model for the multi-stage batch scheduling problem is derived from constraints (2.1)-(2.3), (2.10), (2.15), and (2.25).

$$
\begin{aligned}
\max \quad & \sum_{i \in T} \sum_{\substack{v \in W \\ v=(o,l,u)}} 0.2 \cdot l \cdot E_{iv} & & \\
\text{s.t.} \quad & D_{iv} = D_v \cdot Z_{iv} & & i \in T, v \in W \\
& E_{iv} \leq \overline{E_v} \cdot Z_{iv} & & i \in T, v \in W \\
& E_{iv} = S_{iv} + D_{iv} & & i \in T, v \in W \\
& \sum_{i \in T} \sum_{v \in W_o} Z_{iv} = 1 & & o \in O \\
& \underline{N_u} \leq \sum_{i \in T} \sum_{v \in W_u} Z_{iv} \leq \overline{N_u} & & u \in U \\
& \sum_{i \in T} \sum_{v \in W_{o(l-1)}} E_{iv} \leq \sum_{i \in T} \sum_{v \in W_{ol}} S_{iv} & & o \in O, l \in L, l \neq 1 \\
& S_{iv} = t_i \cdot Z_{iv} & & i \in T, v \in W \\
& \sum_{v \in W_u} \left\{ Z_{iv} + \sum_{\substack{j \in T \\ i - \delta_u(v) < j < i}} Z_{jv} \right\} \leq 1 & & i \in T, u \in U, \delta_u(v) = \left\lceil \frac{D_v + TR_u}{H/n} \right\rceil \\
& S_{iv}, D_{iv}, E_{iv} \geq 0 & & i \in T, v \in W \\
& Z_{iv} \in \{0,1\} & & i \in T, v \in W \\
& t_i = \frac{i-1}{n} \cdot H & & i \in T
\end{aligned}
$$

To solve this model, the multiplicative approach is used and the initial number of priority-slots is set to $n_0 = H/4 = 125$. We use a time limit of 1,000s as a stopping criterion. Computational results are given in Table 2.11. As for the single-stage case, the MOS-FST time representation quickly finds near-optimal solutions in the first iterations. The instance with 10 orders was not solved due to memory limitations.

2.8.4 Models Comparison

Figure 2.11 shows how the objective value of the best incumbent varies over time. From this figure, it is clear that the MOS model is superior to the other models as it finds first feasible solutions very fast and is able to find optimal solutions in reasonable time. The MOS-FST model is significantly more expensive than previous single-stage cases as the scheduling

Table 2.11: MOS-FST computational results for multi-stage batch scheduling problems.

Pb	n	LP	MILP	Nb of nodes	CPU	Cumulative CPU
MSBSP5	125	6,820.61	6,800.60	0	16.84s	16.84s
	250	6,826.21	6,819.00	0	39.32s	56.16s
	500	6,827.76	6,824.80	499	155.83s	211.99s
	1,000	6,828.76	6,828.00	509	619.69s	831.68s
	2,000	6,828.76	6,828.10	0	+1,000s	+1,831.68s
MSBSP8	125	10,951.49	10,925.80	503	71.54s	71.54s
	250	10,965.66	10,956.60	31,692	+1,000s	+1,071.54s

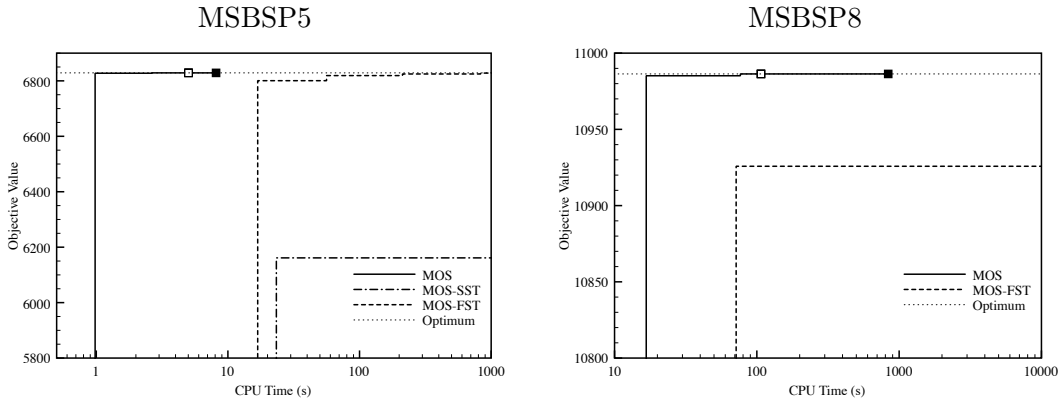


Figure 2.11: Comparison of time representations for multi-stage batch scheduling problems.

horizon is much longer (500 hours as opposed to 30 hours). Therefore, more priority-slots need to be postulated which makes the model size grow significantly. Janak et al. (2004), Castro et al. (2006) and Liu and Karimi (2007) have developed and studied several models that can be applied to different types of multi-stage batch scheduling problems. Their computational results show similar trends.

2.9 Conclusion

In this chapter, we have presented four different time representations that in different forms have been previously introduced to solve process scheduling problems. Using the common concept of priority-slot, it was shown that it is possible to derive relationship results between these time representations. Additionally, generic scheduling constraints were presented with

corresponding strengthened formulations that rely on exploiting the non-overlapping graph structure of these problems through maximum cliques and bicliques. These formulations were developed as well as specific solution algorithms for each model. Given the unifying framework of these models, we were able to apply them to batch scheduling of single and multi-stage plants.

Intuitively, the representation that requires the fewest number priority-slots is the most computationally efficient. In practice, the MOS time representation proved to be superior to the other time representations on the two types of problem tested due to the smaller number of priority-slots and symmetry-breaking constraints used.

In terms of computational performance, the discrete-time MOS-FST representation is comparable to the MOS time representation due to the high efficiency of constraint (2.25) for problems with constant processing times and time horizon of reasonable length.

On the other hand, the MOS-SST time representation performs poorly as it requires many more priority-slots than the MOS representation but does not improve the LP relaxation of the MILP model.

Finally, it should be noted that precedence-based formulations (Méndez et al., 2001) are very good alternatives to priority-slot based representations. These formulations tend to be limited to the context of batch scheduling problems without inventory tracking (Méndez et al., 2006a). The corresponding models often have weak LP relaxations as they require a large proportion of big-M constraints, but they are very compact and therefore easily solved by current commercial MILP solvers.

Chapter 3

Short-Term Scheduling of Crude-Oil Operations

3.1 Introduction

The optimal scheduling of crude-oil operations has been studied since the 90's and has been shown to lead to multimillion dollar benefits by Kelly and Mann (2003) as it is the first stage of the oil refining process. It involves crude-oil unloading from crude marine vessels (at berths or jetties) or from a pipeline to storage tanks, transfers from storage tanks to charging tanks and atmospheric distillations of crude-oil mixtures from charging tanks. The crude is then processed in order to produce basic products which are then blended into gasoline, diesel, and other final products. Assuming that the schedule of crude supply and production demands are determined by the long-term refinery planning, this chapter studies the short-term scheduling problem maximizing gross margins of crude-oil mixtures.

Shah (1996) proposed to use mathematical programming techniques to find crude-oil schedules exploiting opportunities to increase economic benefits. Lee et al. (1996) considered a crude-oil scheduling problem involving crude unloading at berths, developed a discrete-time MINLP model, and solved an MILP relaxation of the model. Later, Wenkay et al. (2002) improved the model and proposed an iterative approach to solve the MINLP model, taking into account the nonlinear blending constraints. Pinto et al. (2000), Moro and Pinto (2004), and Reddy et al. (2004) used a global event formulation to model refinery systems involving crude-oil unloading from pipeline or jetties. The scheduling horizon is divided into fixed length sub-intervals, which are then divided in several variable length

time-slots. In parallel, Jia et al. (2003) developed an operation-specific event model and applied it to the problems introduced by Lee et al. (1996) using a linear approximation of storage costs. A comparison of computational performances between both continuous-time and discrete-time models was given showing significant decreases in CPU time. Also, solutions that are not guaranteed to be globally optimal were obtained using standard MINLP algorithms. Recently, Furman et al. (2007) presented a more accurate version of the event-point formulation, and Karuppiah et al. (2008) later addressed the global optimization of this model using an outer-approximation algorithm where the MILP master problem is solved by a Lagrangean decomposition. While rigorous, this method can be computationally expensive.

The aim of this chapter is to apply different time representations and mathematical models to the crude-oil scheduling problem introduced by Lee et al. (1996). First, the problem definition is given. Next, the MOS, MOS-SST, and MOS-FST models (see chapter 2) are derived and a simple solution method to solve them is presented. Finally, computational results are given to show the effectiveness of the proposed model and solution method. As it requires additional work to break its inherent symmetries, the SOS model will be detailed in chapter 4.

3.2 Problem Statement

3.2.1 General Description

This work is aimed at solving the four examples of refinery crude-oil operations problems introduced in Lee et al. (1996), respectively denoted COSP1, ..., COSP4. Each crude-oil operations system is composed of four types of resources: crude marine vessels, storage tanks, charging tanks and crude distillation units (CDUs). Three types of operations, all transfers between resources, are allowed: crude-oil unloadings from marine vessels to storage tanks, transfers between tanks, and transfers from charging tanks to CDUs.

A crude-oil scheduling problem is defined by: (a) a time horizon, (b) arrival time of marine

vessels, (c) capacity limits of tanks, (d) transfer flowrate limitations, (e) initial composition of vessels and tanks, (f) crude property specifications for distillations, (g) and demands for each crude blend.

The logistics constraints of the problem are defined as follows.

(i) Only one berth is available at the docking station for vessel unloadings,
(ii) simultaneous inlet and outlet transfers on tanks are forbidden,
(iii) a tank may charge only one CDU at a time,
(iv) a CDU can be charged by only one tank at a time,
(v) and CDUs must be operated continuously throughout the scheduling horizon.

In this chapter, the goal is to determine how many times each operation will be executed, when it will be performed (start time and duration), and the volume of crude to be transferred in order to maximize the gross margins of distilled mixed oil. The gross margin of each crude can be estimated from the sales income of the final products minus its purchase value and related refining operational costs. Depending on the market value of the different crudes and final products, the model tries to process the most profitable crudes and to store the other crudes. As opposed to the work of Lee et al. (1996), sea waiting, unloading, storage and CDU switching costs are ignored.

As CDU switches between different crude blends are costly, it is considered that the number of distillation operations is bounded. The bounds on the number of distillation operations can be set by the user, or for the lower bound, it can be obtained by solving a model minimizing the number of distillation operations. Typically, the number and type of marine vessels is determined at the planning level as well as demands in each crude blend.

Due to the logistics constraint (ii), inventory tracking in tanks can be performed by looking at the sequence of inlet and outlet operations only, and not at the sequence of start and end events of such operations. Indeed, for each tank, the scheduling solution can be decomposed into successions of inlet and outlet states during which one or several operations are performed. Therefore, it is only necessary to enforce capacity limitations

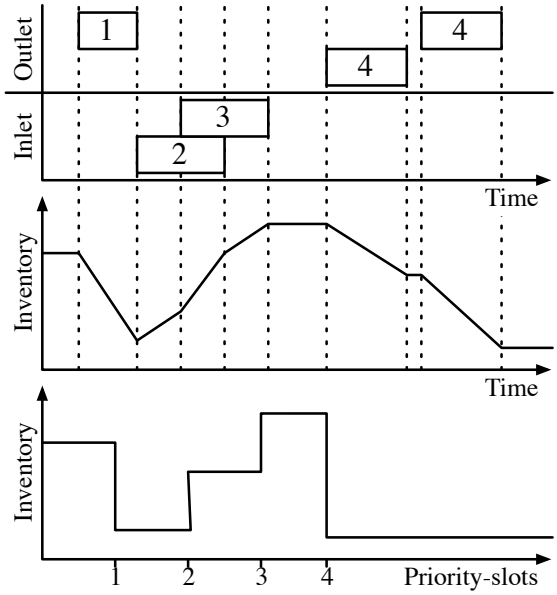


Figure 3.1: Example of tank schedule.

at the transitions between these states. Figure 3.1 depicts an example of tank schedule with inlet and outlet states, time-based and priority-slot-based inventory profiles. Each transfer activity is represented by a horizontal bar labeled with the corresponding priority-slot. Note that the two inventory profiles do not coincide due to the use of different x-axes. In particular, in the priority-slot-based profile, it is not possible to distinguish the two consecutive inventory decreases corresponding to the two operations assigned to priority-slot 4. Under assumption (ii), it is sufficient to enforce tank capacity limitations just before transition slots as it corresponds to inventory upper and lower peaks. In practice, they are enforced just before all priority-slots.

3.2.2 Case Study

Figure 1.4 depicts the refinery configuration for COSP1. The scheduling horizon is composed of 8 days, and two marine vessels are scheduled to arrive at the beginning of day 1 ($t = 0$) and day 5 ($t = 4$), and contain 1 million bbl of crude-oil A and B, respectively. There is one CDU which has to process 1 million bbl of each crude-oil mixture, X and Y. The property

Table 3.1: Data for COSP1.

Scheduling horizon			8 days
Vessels	Arrival time	Composition	Amount of crude (Mbbl)
Vessel 1	0	100% A	1,000
Vessel 2	4	100% B	1,000
Storage tanks	Capacity (Mbbl)	Initial composition	Initial amount of crude (Mbbl)
Tank 1	[0, 1,000]	100% A	250
Tank 2	[0, 1,000]	100% B	750
Charging tanks	Capacity (Mbbl)	Initial composition	Initial amount of crude (Mbbl)
Tank 1 (mix X)	[0, 1,000]	100% C	500
Tank 2 (mix Y)	[0, 1,000]	100% D	500
Crudes	Property 1 (sulfur concentration)		Gross margin ($/bbl)
Crude A	0.01		9
Crude B	0.06		4
Crude C	0.02		8
Crude D	0.05		5
Crude mixtures	Property 1 (sulfur concentration)		Demand (Mbbl)
Crude mix X	[0.015, 0.025]		[1,000, 1,000]
Crude mix Y	[0.045, 0.055]		[1,000, 1,000]
Unloading flowrate	[0, 500]	Transfer flowrate	[0, 500]
Distillation flowrate	[50, 500]	Number of distillations	3

that is being tracked is the weight fraction of sulfur, 0.01 for crude A and 0.06 for crude B. The sulfur concentration of crude mix X and Y should be in the ranges $[0.015, 0.025]$ and $[0.045, 0.055]$, respectively. The two storage tanks initially contain 250,000 bbl of crude A and 750,000 bbl of crude B. The two charging tanks initially contain 500,000 bbl of crude C and D, with sulfur concentrations of 0.02 and 0.05, respectively. Crudes C and D are in fact blends of crudes A and B that match the sulfur concentration specifications for crude mix X and Y. The gross margins of crudes A, B, C, and D are 9 $/bbl, 4 $/bbl, 8 $/bbl, 5 $/bbl, respectively. Gross margins for crudes C and D have been calculated from gross margins of crudes A and B. The data for COSP1 is given in Table 3.1. Flowrate limitations are expressed in Mbbl/day.

Figure 3.2 depicts the Gantt chart of a sub-optimal solution for COSP1 with a profit of $6,925,000 that might be obtained by using heuristics. Each task is represented by a horizontal bar and each row corresponds to a specific operation. Tank inventories are also displayed, showing that tank capacity limits are satisfied. Figure 3.3 shows in contrast the Gantt chart of the optimal solution obtained, and proved optimal (i.e. 0% optimality gap),

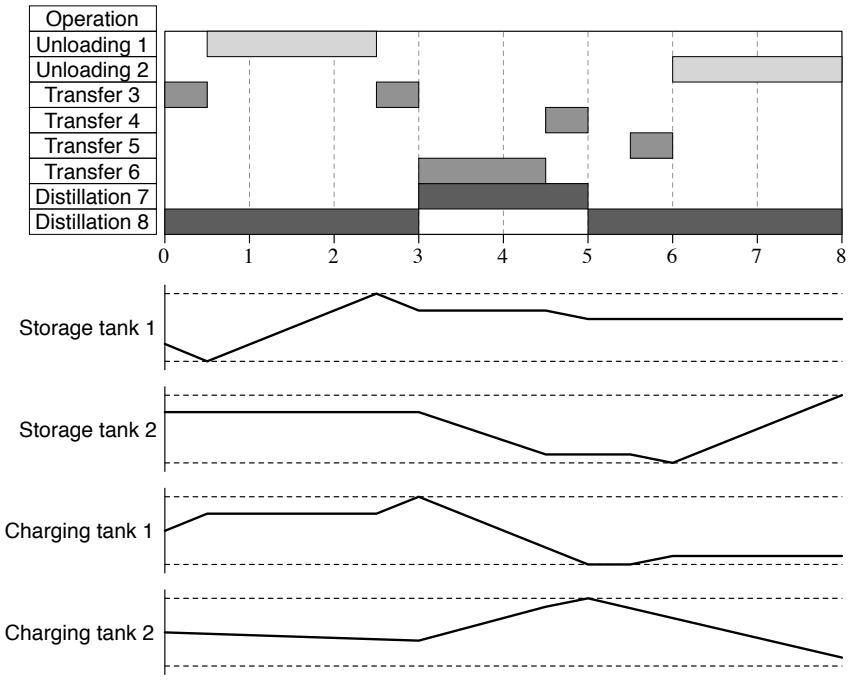


Figure 3.2: Sub-optimal schedule for COSP1 (profit: \$6,925,000).

with the proposed approach as shown in section 3.5. It corresponds to a profit of \$7,975,000, which represents a 13.2% increase. Clearly, finding such a solution is non-trivial.

3.3 Mathematical Models

In this section, we present three mathematical models based on the MOS, MOS-SST, and MOS-FST time representations (see chapter 2). They are all based on the same sets, parameters and variables.

3.3.1 Sets

The following sets will be used in the model.

- $T = \{1, \ldots, n\}$ is the set of priority-slots
- W is the set of all operations: $W = W_U \cup W_T \cup W_D$ ($W = \{1..8\}$ for COSP1, see Figure 1.4)

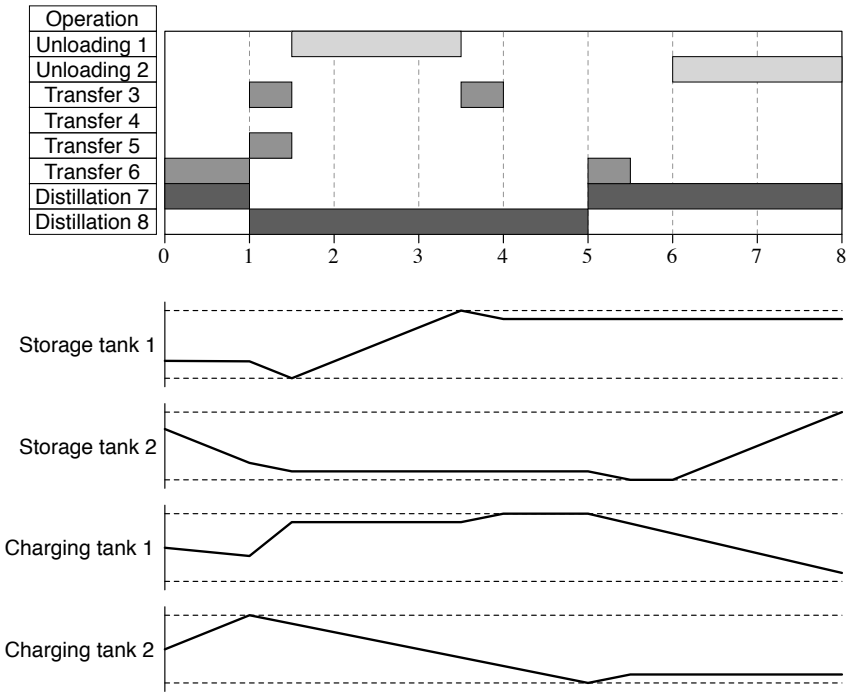


Figure 3.3: Optimal schedule for COSP1 (profit: \$7,975,000).

- $W_U \subset W$ is the set of unloading operations ($W_U = \{1, 2\}$ for COSP1)
- $W_T \subset W$ is the set of tank-to-tank transfer operations ($W_T = \{3, 4, 5, 6\}$ for COSP1)
- $W_D \subset W$ is the set of distillation operations ($W_D = \{7, 8\}$ for COSP1)
- R is the set of resources (i.e. tanks, units): $R = R_V \cup R_S \cup R_C \cup R_D$
- $R_V \subset R$ is the set of vessels
- $R_S \subset R$ is the set of storage tanks
- $R_C \subset R$ is the set of charging tanks
- $R_D \subset R$ is the set of distillation units
- $I_r \subset W$ is the set of inlet transfer operations on resource r
- $O_r \subset W$ is the set of outlet transfer operations on resource r
- C is the set of products (i.e. crudes)
- K is the set of product properties (e.g. crude sulfur concentration)

3.3.2 Parameters

The following parameters are given.

- H is the scheduling horizon
- $[\underline{V_v^t}, \overline{V_v^t}]$ are bounds on the total volume transferred during transfer operation v ; in all instances, $\underline{V_v^t} = 0$ for all operations except unloadings for which $\underline{V_v^t} = \overline{V_v^t}$ is the volume of crude in the marine vessel
- $[\underline{N_D}, \overline{N_D}]$ are the bounds on the number of distillations
- $[\underline{FR_v}, \overline{FR_v}]$ are flowrate limitations for transfer operation v
- $\underline{S_v}$ is the minimum start time of unloading operation $v \in W_U$ (i.e. arrival time of the corresponding vessel)
- $[\underline{x_{vk}}, \overline{x_{vk}}]$ are the limits of property k of the blended products transferred during operation v
- x_{ck} is the value of the property k of crude c
- $[\underline{L_r^t}, \overline{L_r^t}]$ are the capacity limits of tank r
- L_{0r}^t is the initial total level in tank r
- L_{0rc} is the initial crude level in tank r for crude c
- $[\underline{D_r}, \overline{D_r}]$ are the bounds of the demand on products to be transferred out of the charging tank r during the scheduling horizon
- G_c is the individual gross margin of crude c

3.3.3 Variables

The variables used in the model are composed of binary assignment variables, and continuous time, operation and resource variables.

- **Assignment variables** $Z_{iv} \in \{0, 1\} \qquad i \in T, v \in W$

  $Z_{iv} = 1$ if operation v is assigned to priority-slot i, $Z_{iv} = 0$ otherwise.
- **Time variables** $S_{iv} \geq 0, D_{iv} \geq 0, E_{iv} \geq 0 \qquad i \in T, v \in W$

  S_{iv} is the start time of operation v if it is assigned to priority-slot i, $S_{iv} = 0$ otherwise.

D_{iv} is the duration of operation v if it is assigned to priority-slot i, $D_{iv} = 0$ otherwise.

E_{iv} is the end time of operation v if it is assigned to priority-slot i, $E_{iv} = 0$ otherwise.

- **Operation variables** $V_{iv}^t \geq 0$ and $V_{ivc} \geq 0$ $\quad i \in T, v \in W, c \in C$

  V_{iv}^t is the total volume of crude transferred during operation v if it is assigned to priority-slot i, $V_{iv}^t = 0$ otherwise.

  V_{ivc} is the volume of crude c transferred during operation v if it is assigned to priority-slot i, $V_{ivc} = 0$ otherwise.

- **Resource variables** L_{ir}^t and L_{irc} $\quad i \in T, r \in R, c \in C$

  L_{ir}^t is the total *accumulated* level of crude in tank $r \in R_S \cup R_C$ before the operation assigned to priority-slot i.

  L_{irc} is the *accumulated* level of crude c in tank $r \in R_S \cup R_C$ before the operation assigned to priority-slot i.

It should be noted that the crude composition of blends in tanks is tracked instead of their properties. The distillation specifications are later enforced by calculating a posteriori the properties of the blend in terms of its composition. For instance, in problem COSP1, a blend composed of 50% of crude A and 50% of crude B has a sulfur concentration of 0.035 which does not meet the specification for crude mix X nor for crude mix Y.

3.3.4 Objective Function

The objective is to maximize the gross margins of the distilled crude blends. Using the individual gross margins G_c, it is written as follows.

$$\max \sum_{i \in T} \sum_{r \in R_D} \sum_{v \in I_r} \sum_{c \in C} G_c \cdot V_{ivc}$$

3.3.5 General Constraints

In this section, we present a set of general constraints which can be used in any of the time representations studied in chapter 2.

The following variable bound and time constraints (3.1) are used.

$$S_{iv} \geq \underline{S_v} \cdot Z_{iv} \qquad i \in T, v \in W_U \tag{3.1a}$$

$$E_{iv} \leq H \cdot Z_{iv} \qquad i \in T, v \in W \tag{3.1b}$$

$$E_{iv} = S_{iv} + D_{iv} \qquad i \in T, v \in W \tag{3.1c}$$

The following unloading and distillation cardinality constraints (3.2) are used.

$$\sum_{i \in T} \sum_{v \in O_r} Z_{iv} = 1 \qquad r \in R_V \tag{3.2a}$$

$$\underline{N_D} \leq \sum_{i \in T} \sum_{v \in W_D} Z_{iv} \leq \overline{N_D} \tag{3.2b}$$

The following unloading precedence constraints (3.3) are used to make sure that crude vessels unload their content according to their respective order of arrival at the refinery. The notation $r_1 < r_2$ denotes that vessel r_1 is scheduled to arrive at the refinery before vessel r_2.

$$\sum_{i \in T} \sum_{v \in O_{r_1}} E_{iv} \leq \sum_{i \in T} \sum_{v \in O_{r_2}} S_{iv} \qquad r_1, r_2 \in R_V, r_1 < r_2 \tag{3.3a}$$

$$\sum_{\substack{j \in T \\ j < i}} \sum_{v \in O_{r_1}} Z_{jv} \geq \sum_{\substack{j \in T \\ j \leq i}} \sum_{v \in O_{r_2}} Z_{jv} \qquad i \in T, r_1, r_2 \in R_V, r_1 < r_2 \tag{3.3b}$$

The following constraint (3.4) states that each CDU must be operated without interruption throughout the scheduling horizon. As CDUs perform only one operation at a time, the continuous operation constraint is defined by equating the sum of the duration of distillations to the time horizon.

$$\sum_{i \in T} \sum_{v \in I_r} D_{iv} = H \qquad r \in R_D \tag{3.4}$$

The following variable constraints (3.5) are directly derived from the definition of volume and level variables.

$$V_{iv}^t \leq \overline{V_v^t} \cdot Z_{iv} \qquad i \in T, v \in W \qquad (3.5a)$$

$$V_{iv}^t \geq \underline{V_v^t} \cdot Z_{iv} \qquad i \in T, v \in W \qquad (3.5b)$$

$$V_{iv}^t = \sum_{c \in C} V_{ivc} \qquad i \in T, v \in W \qquad (3.5c)$$

$$L_{ir}^t = L_{0r}^t + \sum_{j \in T, j<i} \sum_{v \in I_r} V_{iv}^t - \sum_{j \in T, j<i} \sum_{v \in O_r} V_{iv}^t \qquad i \in T, r \in R \qquad (3.5d)$$

$$L_{irc} = L_{0rc} + \sum_{j \in T, j<i} \sum_{v \in I_r} V_{ivc} - \sum_{j \in T, j<i} \sum_{v \in O_r} V_{ivc} \qquad i \in T, r \in R, c \in C \qquad (3.5e)$$

$$L_{ir}^t = \sum_{c \in C} L_{irc} \qquad i \in T, r \in R \qquad (3.5f)$$

The following operation constraints (3.6) include:

1. flowrate limitations that link volume and duration variables
2. property specifications, assuming that the mixing rule is linear
3. composition constraints, which are nonlinear

$$\underline{FR_v} \cdot D_{iv} \leq V_{iv}^t \leq \overline{FR_v} \cdot D_{iv} \qquad i \in T, v \in W \qquad (3.6a)$$

$$\underline{x_{vk}} \cdot V_{iv}^t \leq \sum_{c \in C} x_{ck} V_{ivc} \leq \overline{x_{vk}} \cdot V_{iv}^t \qquad i \in T, v \in W, k \in K \qquad (3.6b)$$

$$V_{ivc} \cdot L_{ir}^t = L_{irc} \cdot V_{iv}^t \qquad i \in T, r \in R, v \in O_r, c \in C \qquad (3.6c)$$

It has been shown (Quesada and Grossmann, 1995b) that processes including both mixing and splitting of streams cannot be expressed as a linear model. Mixing occurs when two streams are used to fill a tank and is expressed linearly in constraints (3.5d-3.5e). Splitting occurs when partially discharging a tank, resulting in two parts: the remaining content of the tank and the transferred products. This constraint is nonlinear. The composition of the products transferred during a transfer operation must be identical to the composition of the origin tank. Note that constraint (3.6c) is a bilinear reformulation of the original constraint (3.7) and is correct even when operation v is not assigned to priority-slot i, as

then $V_{iv}^t = V_{ivc} = 0$.

$$\frac{L_{irc}}{L_{ir}^t} = \frac{V_{ivc}}{V_{iv}^t} \qquad i \in T, r \in R, v \in O_r, c \in C \tag{3.7}$$

The following resource constraints (3.8) models inventory capacity limitations. As simultaneous charging and discharging of tanks is forbidden, these constraints are sufficient.

$$\underline{L_r^t} \leq L_{ir}^t \leq \overline{L_r^t} \qquad i \in T, r \in R_S \cup R_C \tag{3.8a}$$

$$0 \leq L_{irc} \leq \overline{L_r^t} \qquad i \in T, r \in R_S \cup R_C, c \in C \tag{3.8b}$$

$$\underline{L_r^t} \leq L_{0r}^t + \sum_{i \in T} \sum_{v \in I_r} V_{iv}^t - \sum_{i \in T} \sum_{v \in O_r} V_{iv}^t \leq \overline{L_r^t} \qquad r \in R_S \cup R_C \tag{3.8c}$$

$$0 \leq L_{0rc} + \sum_{i \in T} \sum_{v \in I_r} V_{ivc} - \sum_{i \in T} \sum_{v \in O_r} V_{ivc} \leq \overline{L_r^t} \qquad r \in R_S \cup R_C, c \in C \tag{3.8d}$$

The following demand constraints (3.9) defines lower and upper limits, $\underline{D_r}$ and $\overline{D_r}$, on the total volume of products transferred out of each charging tank r during the scheduling horizon.

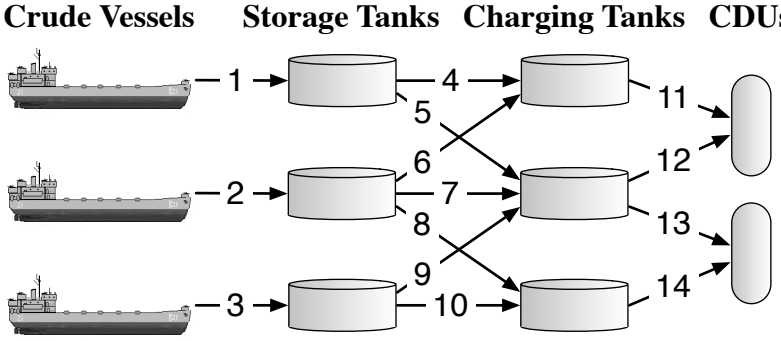
$$\underline{D_r} \leq \sum_{i \in T} \sum_{v \in O_r} V_{iv}^t \leq \overline{D_r} \qquad r \in R_C \tag{3.9}$$

3.3.6 Strengthened Constraints

Figure 3.4 displays the refinery system for problems COSP2 and COSP3. Each labelled arc corresponds to a transfer operation. Figure 3.5 displays the corresponding non-overlapping graph. It contains 4 maximal cliques of 3 operations: $\{1, 2, 3\}$, $\{5, 12, 13\}$, $\{7, 12, 13\}$, and $\{9, 12, 13\}$. In particular, maximal clique $\{5, 12, 13\}$ is due to non-overlapping constraints (ii) and (iii) which makes it non trivial to detect. Therefore, a generic maximal clique finding algorithm allows generating non-trivial strengthened non-overlapping constraints, which is the main objective of the non-overlapping graph representation.

The maximum cliques of G_{NO} are used to derive the following assignment and scheduling

Figure 3.4: Refinery crude-oil scheduling system for problem COSP2 and COSP3.



constraints, which are identical to constraints (2.17) and (2.19).

$$\sum_{v \in W'} Z_{iv} \leq 1 \qquad i \in T, W' \in clique(G_{NO}) \tag{3.10}$$

$$\sum_{v \in W'} E_{i_1 v} + \sum_{\substack{i \in T \\ i_1 < i < i_2}} \sum_{v \in W'} D_{iv} \leq \sum_{v \in W'} S_{i_2 v} + H \cdot (1 - \sum_{v \in W'} Z_{i_2 v}) \qquad i_1, i_2 \in T, i_1 < i_2, W' \in clique(G_{NO}) \tag{3.11}$$

3.3.7 Symmetry-Breaking Constraint for MOS Models

As mentioned by Kallrath (2002), degeneracies and symmetries often cause scheduling problems to be difficult to solve. The author suggests that nonlinearities involved in refinery scheduling problems may reduce these effects. In this work, we consider the concept of static symmetry-breaking constraints as presented by Margot (2008) in order to break the symmetries that can be detected in the MOS model. In particular, the following generic constraint is used. It states that an operation v cannot be assigned to priority-slot i if no other non-overlapping operation is assigned to priority-slot $i-1$. Indeed, in this case operation v can be assigned to slot $i-1$ instead of slot i with no impact on the scheduling solution.

$$Z_{iv} \leq \sum_{\substack{v' \in W \\ NO_{vv'}=1}} Z_{(i-1)v'} \qquad i \in T, i > 1, v \in W \tag{3.12}$$

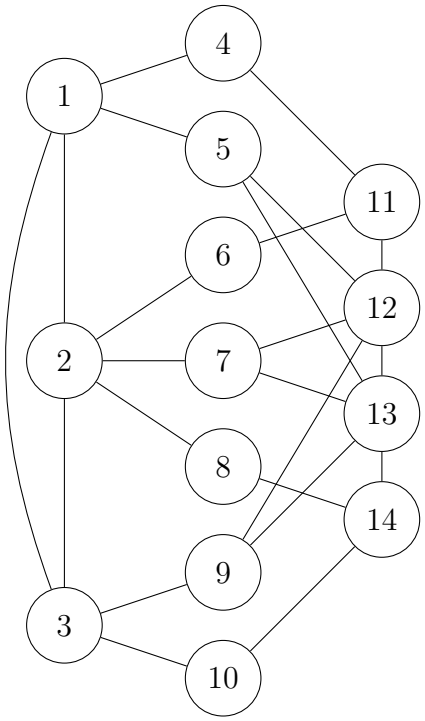


Figure 3.5: Non-overlapping graph for crude-oil examples 2 and 3.

This constraint does not make any assumptions on the characteristics of the problem as opposed to symmetry-breaking constraint (2.29) which is specific to the case of batch scheduling problems.

3.3.8 Full Models

The full mathematical formulations for each time representation presented in chapter 2 are described in appendix C. The details for the SOS model are given in chapter 4.

3.4 Solution Method

The non-convex MINLP models described in the previous section can be solved using any generic MINLP solver such as DICOPT (outer-approximation method) or BARON (global solver using a spatial branch-and-bound search). The former code may converge to a poor

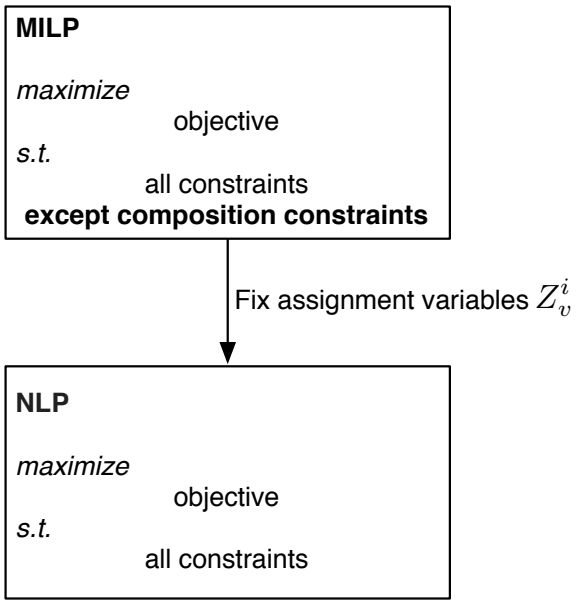


Figure 3.6: Two step decomposition strategy.

sub-optimal solution, while the latter can be prohibitively expensive to use. Therefore, a simple two-step procedure has been implemented (see Figure 3.6), leading to local optimal solutions with an estimation of the optimality gap.

In the first step, a linear MILP relaxation of the model obtained after dropping the nonlinear composition constraint (3.6c) is solved. It should be noted that material balances are still enforced in this MILP relaxation through constraints (3.5c-3.5f). Furthermore, the number of priority-slots n is selected at this stage using the additive or multiplicative approach as described in section 2.6.

The solution returned by the MILP solver may violate the bilinear composition constraint (3.6c). In this case, the binary variables Z_{iv} are fixed, which means that the sequence of operations is fixed, and the resulting nonlinear programming (NLP) model is solved using the solution of the MILP as a starting point. This NLP model contains the same constraints as in the MILP model plus the nonlinear constraint (3.6c). The solution obtained at this stage might not be the optimum of the full model, but the optimality gap can be estimated from the upper bound given by the MILP solution and the lower bound given by the NLP

solution.

The NLP model can be solved using either a global NLP solver (e.g. BARON) or a local NLP solver (e.g. CONOPT). In the former case, the solver will return the global optimum of the NLP if it is feasible. One could then add integer cuts to the MILP in order to close the gap between the upper level and the lower level of the decomposition. This method will return the global optimum of the MINLP. In the latter case, the NLP being non-convex, several different locally optimal solution may be obtained depending on the starting point. Thus, there is no proof that the iterative procedure will return the global optimum or even a feasible solution to the MINLP, unless the NLP returns an objective value identical to the MILP's in which case global optimality is proved. As shown in section 3.5, the solutions obtained at the first iteration are near-optimal so the procedure has been stopped at this stage.

3.5 Computational Results

All experiments have been performed on an Intel Xeon 1.86GHz processor with GAMS as modeling language, CPLEX 11 as MILP solver and CONOPT 3 as local NLP solver. The CPU limit for solving the MILP has been set to 1,000s. The scheduling systems and data for the four crude-oil operations scheduling problems from Lee et al. (1996) are given in appendix B.

3.5.1 Scheduling Results

Figures 3.7 and 3.8 depicts the solutions obtained for problems COSP2 and COSP3 using the proposed approach. The solution for COSP2 is proved optimal while the solution for COSP3 is found within a 2.3% optimality gap. As there is no incentive to keep inventory in the charging tanks, their final level is in general close to zero. However, the crude-oil that arrived late to the refinery is mostly kept in the storage tanks. This leads to higher transfer activity at the beginning than at the end of the scheduling horizon.

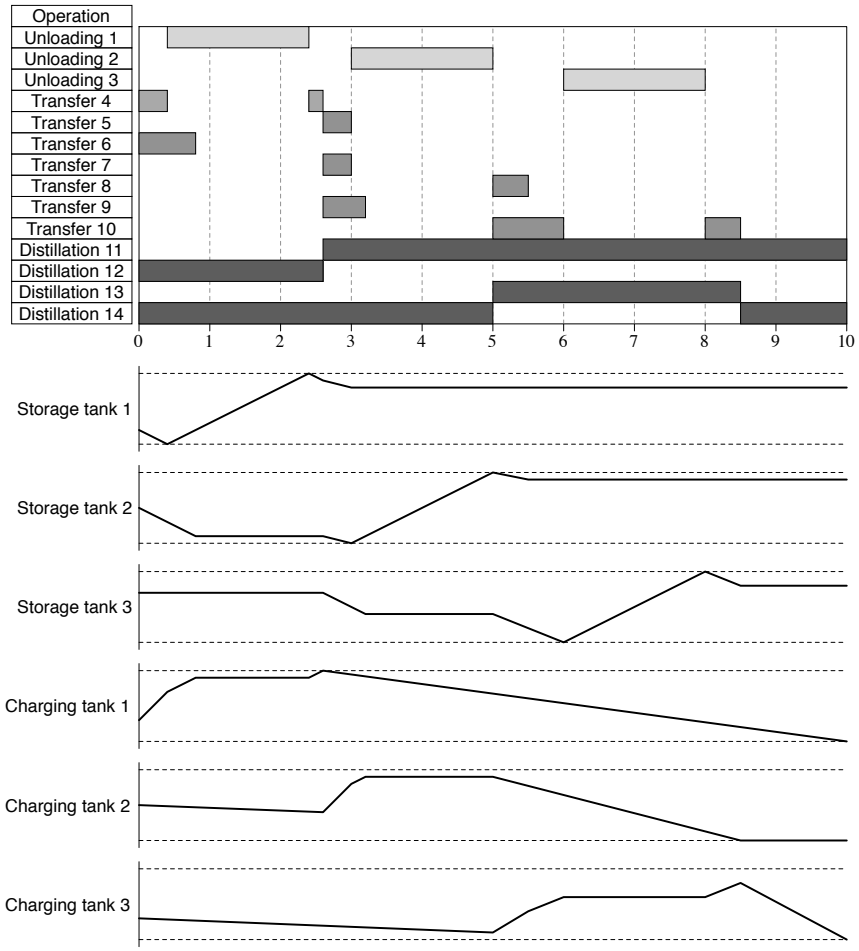


Figure 3.7: Optimal schedule for COSP2 (profit: \$10,117,000).

The main uncertain parameter in refinery crude-oil scheduling problems is the arrival time of tankers. The expected date of arrival of these marine vessels is known long in advance, but is also subject to many changes before it actually arrives at the refinery. Figure 3.9 shows the optimal schedule for COSP2 when vessels are scheduled to arrive one day later, leading to a 3.4% profit decrease (from \$10,117,00 to \$9,775,000). It should be noted that the sequence of distillation operations is different in this case, and that there is a higher transfer activity in the very beginning of the scheduling horizon. However, this schedule assumes that the late arrival of vessels is known at time $t = 0$.

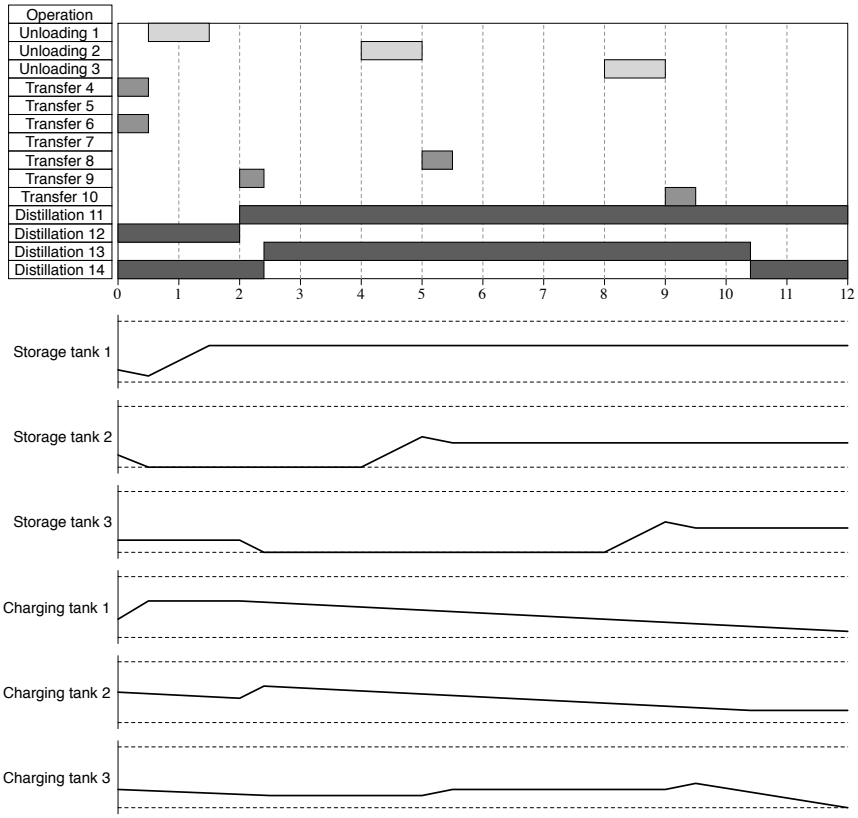


Figure 3.8: Schedule obtained for COSP3 within 2.3% optimality gap (profit: $8,540,000).

Assuming that the exact arrival dates are known slightly later ($t = \varepsilon \ll 1$), the initial decisions are fixed, so that operations 12, 14, 4, and 6 (which start at time $t = 0$ in the original solution) are forced to start at time $t = 0$, but with variable durations. In this case, the optimal solution is depicted in Figure 3.10. This solution is very similar to the original optimal solution (Figure 3.7) as most transfers are simply delayed, although the profit is reduced to $9,609,000. Only transfer operation 10 from $t = 8$ to $t = 8.5$ is no longer used. The fact that the exact arrival of vessels is known only at $t = \varepsilon$ leads to a 1.7% profit decrease (from $9,775,000 to $9,609,000) compared to the case where it is known at $t = 0$. This result shows that the initial decisions taken at $t = 0$ do not lead to a large under-optimization in case of late vessel arrivals.

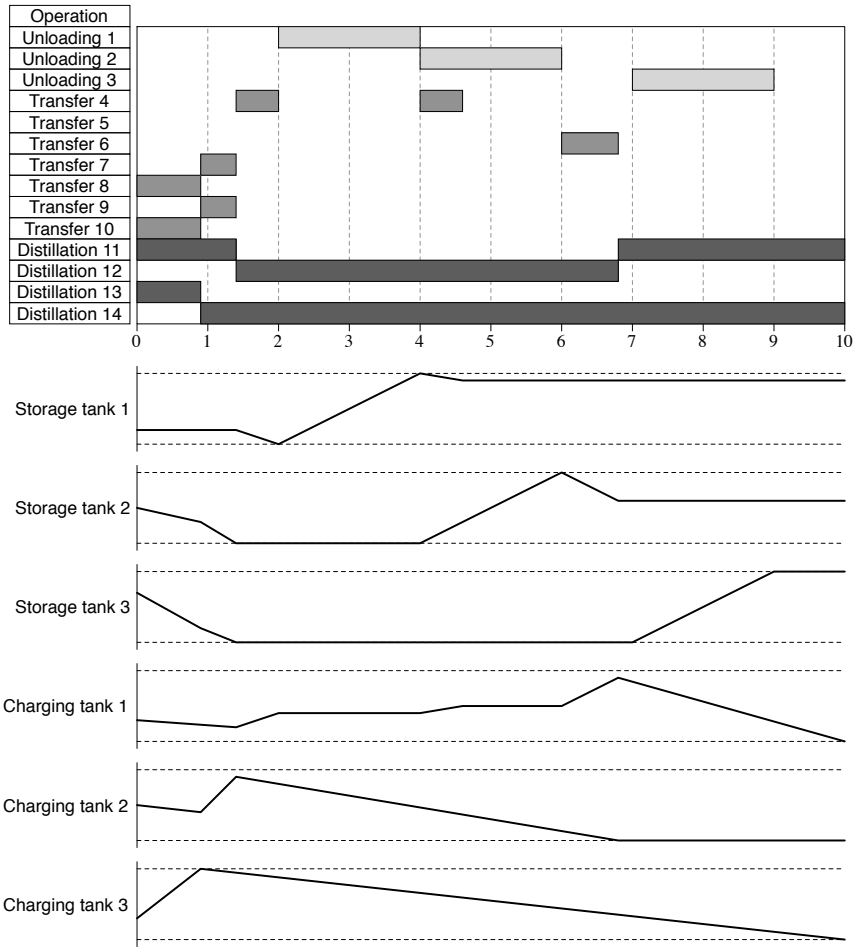


Figure 3.9: Optimal schedule for COSP2 with late vessel arrivals (profit: \$9,775,000).

3.5.2 Performance of the MOS Model

The MILP relaxation of the MOS model is solved with the additive approach (see chapter 2) using the second stop criterion ($\Delta \leq 0$). The initial number of priority-slots is set to $n_0 = 1$. Computational results are given in Table 3.2. All instances are solved within 20 seconds with a few priority-slots. The NLP is modeled using the optimal number of priority-slots determined at the MILP stage and is always solved in less than one second. The most difficult instance, that is COSP2, is the one requiring the most priority-slots although it is not the largest in size. The optimality gap is always small (less than 3%) even though the

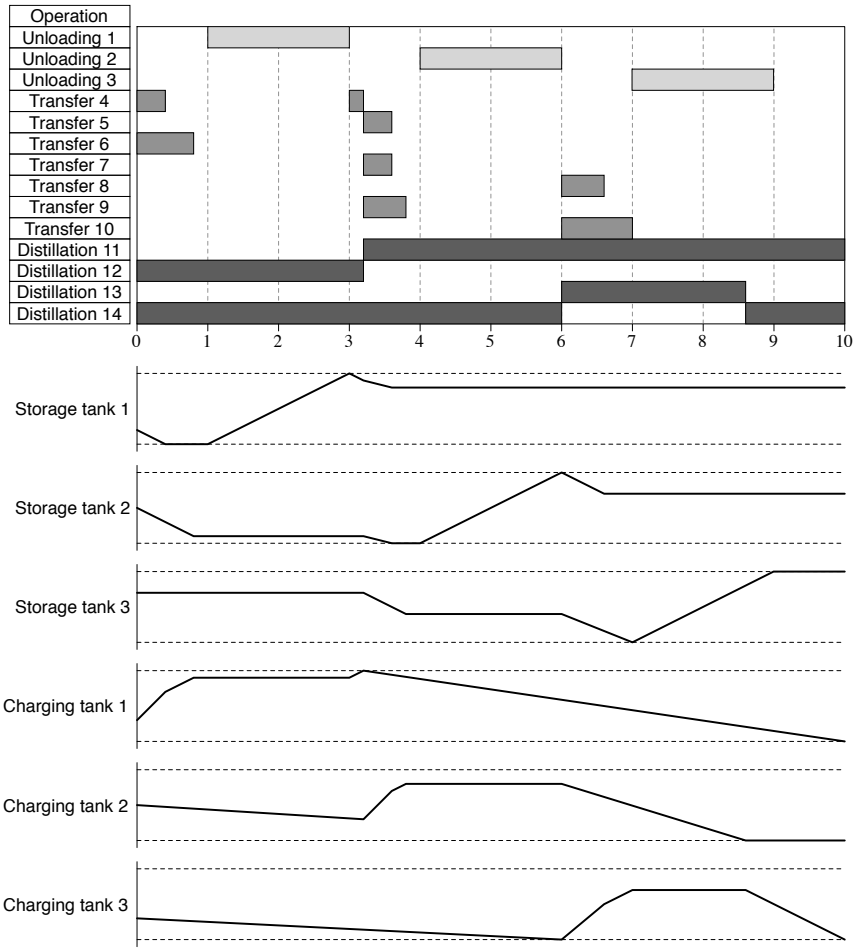


Figure 3.10: Optimal schedule for COSP2 with late vessel arrivals and fixed initial decisions (profit: \$9,609,000).

solution strategy does not necessarily converge to a global optimum. This is explained by the fact that the composition constraints are always satisfied, even if dropped, under specific conditions. For each transfer v assigned to priority-slot i, the composition constraint (3.6c) is satisfied if either of the following conditions holds true.

a) the origin tank contains only one crude before the transfer

b) the origin tank is fully discharged during the transfer

Table 3.2: MOS computational results for crude-oil scheduling problems.

Pb	n	LP	MILP	Number of nodes	CPU	Cumulative CPU	NLP	Gap
	1-4		infeas		3.08s	3.08s		
COSP1	5	80.000	79.750	48	1.16s	4.24s	**79.750**	0%
	6	80.000	no solution	0	1.56s	5.80s		
	1-3		infeas		2.22s	2.22s		
	4	103.000	90.000	0	1.05s	3.27s		
COSP2	5	103.000	96.170	122	2.30s	5.57s		
	6	103.000	101.175	225	3.64s	9.21s	**101.175**	0%
	7	103.000	no solution	335	8.61s	17.82s		
	1-2		infeas		1.43s	1.43s		
	3	84.905	82.500	0	0.85s	2.28s		
COSP3	4	100.000	84.500	16	1.26s	3.54s		
	5	100.000	87.400	63	1.74s	5.28s	**85.400**	2.3%
	6	100.000	no solution	300	3.57s	8.85s		
	1-3		infeas		2.59s	2.59s		
COSP4	4	132.585	132.548	21	1.58s	4.17s	**132.548**	0%
	5	132.585	no solution	0	1.72s	5.89s		

These conditions are always met in the optimal solutions of all problems except COSP3. They do not always hold true in the optimal solution of COSP3, which explains the positive gap obtained.

3.5.3 Performance of the MOS-SST Model

The MILP relaxation of the MOS-SST model is solved with the additive approach using the second stop criterion ($\Delta \leq 0$). The initial number of priority-slots is set to $n_0 = 1$. Computational results are given in Table 3.3. All instances are solved within 400 seconds with slightly more priority-slots than for the MOS model. The most difficult instance is still COSP2. It is interesting to note that the solution obtained for this instance is not actually globally optimal as its objective value (101.174) is slightly lower than 101.175 (see Table 3.2). The actual global optimal solution can be obtained with at least 10 priority-slots. It clearly shows that the second stopping criterion does not guarantee global optimality, although it is very efficient in practice.

Table 3.3: MOS-SST computational results for crude-oil scheduling problems.

Pb	n	LP	MILP	Number of nodes	CPU	Cumulative CPU
COSP1	1-4		infeas		2.97s	2.97s
	5	80.000	79.722	144	1.25s	4.22s
	6	80.000	79.750	288	1.94s	6.16s
	7	80.000	no solution	968	6.30s	12.46s
COSP2	1-4		infeas		3.30s	3.30s
	5	103.000	90.000	327	4.59s	7.89s
	6	103.000	97.726	2316	33.51s	41.40s
	7	103.000	97.751	4672	70.26s	111.66s
	8	103.000	101.174	4545	88.45s	200.11s
	9	103.000	no solution	7007	185.29s	385.40s
COSP3	1-3		infeas		2.52s	2.52s
	4	100.000	82.500	48	1.43s	3.95s
	5	100.000	84.500	333	3.42s	7.37s
	6	100.000	87.000	931	14.83s	22.20s
	7	100.000	87.400	2101	38.02s	60.22s
	8	100.000	no solution	3900	99.03s	159.25s
COSP4	1-5		infeas		24.84s	24.84s
	6	132.585	132.548	1293	28.62s	53.46s
	7	132.585	no solution	500	20.74s	74.20s

3.5.4 Performance of the MOS-FST Model

The MILP relaxation of the MOS-FST model is solved with the multiplicative approach using the second stop criterion ($\Delta \leq 0$). The initial number of priority-slots is set to $n_0 = 4$. Computational results are given in Table 3.4. Only COSP1 is solved to global optimality although feasible solutions are obtained quickly. Within the 1,000 second time limit, near-optimal solutions are obtained for instance COSP3 and COSP4, but the solution obtained for COSP3 shows a 4.8% gap with the best known solution of the MILP relaxation. The MOS-FST discrete-time representation is not efficient at solving the crude-oil scheduling problem. Indeed, as variable processing times are used, constraint (2.24) does not help strengthening the model. Instead, if one wishes to use a discrete-time approach, it would be preferable to consider the work of Lee et al. (1996). Indeed, in their formulation, each operation that is executed over several consecutive time intervals is actually split into several

Table 3.4: MOS-FST computational results for crude-oil scheduling problems.

Pb	n	LP	MILP	Number of nodes	CPU	Cumulative CPU
COSP1	4	80.000	infeas	0	0.87s	0.87s
	8	80.000	79.750	74	1.56s	1.56s
	16	80.000	no solution	100	15.17s	17.60s
COSP2	4	98.000	infeas	0	0.90s	0.90s
	8	103.000	90.000	2114	28.76s	29.66s
	16	103.000	96.250	11852	+1,000s	+1,029.66s
COSP3	4	84.929	82.500	0	0.93s	0.93s
	8	99.120	84.250	22	2.55s	3.48s
	16	100.000	87.156	563	29.97s	33.45s
	32	100.000	no solution	300	+1,000s	+1,033.45s
COSP4	4	132.585	infeas	0	1.20s	1.20s
	8	132.585	132.266	574	12.82s	14.02s
	16	132.585	132.362	2769	376.84s	390.86s
	32	132.585	no solution	0	+1,000s	+1,390.86s

smaller operations, one for each time interval.

3.5.5 Performance of the MILP-NLP Decomposition Strategy

Table 3.5 shows the performance of different solution methods on all four problems using the MOS time representation. MINLP solvers are given as a starting point the solution of the LP obtained by removing the nonlinear constraints of the model and relaxing the integrality constraints on binary variables. Furthermore, the MINLPs are solved with the direct approach using the optimal number of priority-slots. The results show that, while requiring lower CPU times than other solvers, the solution obtained with the two-step procedure is optimal for all problems but COSP3 for which it is very close to the best known solution. The best alternative is to use DICOPT as the iterative outer-approximation algorithm behaves similarly to the MILP-NLP decomposition. Indeed, during the first iteration, DICOPT solves an MILP and then solves the NLP obtained after fixing the binary variables. The main difference comes from the fact that in the first MILP, nonlinear constraints are linearized at the solution of the initial NLP relaxation instead of being

Table 3.5: Performance of different MINLP algorithms for crude-oil scheduling problems.

Solution method	Problem	Slots	Solution	CPU time	Optimality gap
MILP-NLP decomposition[a]	COSP1	1-6	79.750	6s	0%
	COSP2	1-7	101.175	18s	0%
	COSP3	1-6	85.400	9s	2.3%
	COSP4	1-5	132.548	6s	0%
DICOPT[b]	COSP1	5	79.350	11s	-
	COSP2	6	101.175	9s	-
	COSP3	5	85.449	8s	-
	COSP4	4	132.548	8s	-
SBB[b]	COSP1	5	79.350	77s	-
	COSP2	6	101.175	891s	-
	COSP3	5	85.449	778s	-
	COSP4	4	132.548	154s	-
AlphaECP[b]	COSP1	5	79.750	76s	-
	COSP2	6	101.175	25s	-
	COSP3	5	82.904	234s	-
	COSP4	4	132.548	171s	-
BARON[b]	COSP1	5	79.750	69s	0%
	COSP2	6	100.896	+1,000s	2.0%
	COSP3	5	60.000	+1,000s	40.0%
	COSP4	4	132.548	645s	0%

[a] MILP solved using the additive procedure and NLP solves with optimal number of priority-slots

[b] MINLP solved with optimal number of priority-slots

dropped. This explains why DICOPT does not find the optimal solution of COSP1. All the MILPs solved in DICOPT, including the first one, are not relaxations of the non-convex MINLP.

3.6 Conclusion

In this chapter, the crude-oil scheduling problem has been stated and the corresponding MOS, MOS-SST, and MOS-FST mathematical formulations have been derived. The unified modeling approach developed in chapter 2 proved to be very effective as most of the constraints used in all models are identical. These models have been strengthened using

the non-overlapping graph which displays non-trivial cliques. These cliques can be used to generate tightening constraints that might have not been determined without using this systematic approach.

As the models contain nonlinear composition constraints, a two-step MILP-NLP decomposition procedure has been developed in order to solve the MINLP models. The number of priority-slots and the sequence of operations is determined at the first MILP stage and a feasible solution is obtained by solving the second-stage NLP. The approach has the advantage of providing optimal or near-optimal solutions with an estimate of the optimality gap. It performs better than other solvers both in terms of computational performance and optimality of the solution. DICOPT is the best alternative for solving the MINLP model directly.

Chapter 4

Single-Operation Sequencing Model for Crude-Oil Operations Scheduling

4.1 Introduction

In this chapter, we develop the SOS model for the crude-oil operations scheduling problems COSP1, ..., COSP4. It is based on the same sets, parameters and variables as the MOS, MOS-SST, and MOS-FST models derived in chapter 3. The objective function (see section 3.3.4) and general constraints (see section 3.3.5) are identical. We first derive strengthened assignment and non-overlapping constraints by selecting both cliques and bicliques from the non-overlapping graph. Next, we introduce symmetry-breaking constraints for the SOS model that are based on a sequencing rule expressed using a regular language and its corresponding deterministic finite automaton. Various computational results demonstrating the impact of such symmetry-breaking techniques and of the number of priority-slots are presented. Finally, a comparison of all mathematical models for the crude-oil scheduling operations problem is provided. The full SOS mathematical model for the crude-oil operations scheduling problem is given in appendix C.

4.2 Strengthened Constraints

As mentioned in chapter 2, both cliques and bicliques of G_{NO} can be used to generate non-overlapping constraints. Table 4.1 displays all maximal cliques and all maximal bicliques that are not derived from cliques for problems COSP2 and COSP3 (see Fig. 3.5). They are 15 maximal cliques and 15 maximal bicliques which corresponds to $15 + 2 \cdot 15 = 45$

Table 4.1: Maximal cliques and bicliques for COSP2 and COSP3.

Nb of vertices	Maximal cliques
2	$\{1,4\}$, $\{1,5\}$, $\{2,6\}$, $\{2,7\}$, $\{2,8\}$, $\{3,9\}$, $\{3,10\}$, $\{4,11\}$, $\{6,11\}$, $\{8,14\}$, $\{10,14\}$, $\{11,12\}$, $\{13,14\}$
3	$\{1,2,3\}$, $\{5,12,13\}$, $\{7,12,13\}$, $\{9,12,13\}$

Nb of vertices	Maximal bicliques
3	$(\{4\};\{1,4,11\})$, $(\{6\};\{2,6,11\})$, $(\{8\};\{2,8,14\})$, $(\{10\};\{3,10,14\})$
4	$(\{5\};\{1,5,12,13\})$, $(\{7\};\{2,7,12,13\})$, $(\{9\};\{3,9,12,13\})$, $(\{11\};\{4,6,11,12\})$, $(\{14\};\{8,10,13,14\})$
5	$(\{1\};\{1,2,3,4,5\})$, $(\{3\};\{1,2,3,9,10\})$, $(\{5,7,9,12,13\};\{12,13\})$
6	$(\{2\};\{1,2,3,6,7,8\})$, $(\{12\};\{5,7,9,11,12,13\})$, $(\{13\};\{5,7,9,12,13,14\})$

constraints (4.1) and (4.2) which are identical to the constraints (2.19) and (2.26) developed in chapter 2.

$$\sum_{v\in W'} E_{i_1 v} + \sum_{\substack{i\in T \\ i_1<i<i_2}} \sum_{v\in W'} D_{iv} \leq \sum_{v\in W'} S_{i_2 v} + H\cdot(1-\sum_{v\in W'} Z_{i_2 v}) \qquad i_1,i_2\in T, i_1<i_2, W'\in clique(G_{NO}) \tag{4.1}$$

$$\sum_{v\in W_1} E_{i_1 v} \leq \sum_{v\in W_2} S_{i_2 v} + H\cdot(1-\sum_{v\in W_2} Z_{i_2 v}) \qquad \begin{array}{r} i_1,i_2\in T, i_1<i_2, \\ (W_1;W_2)\in biclique(G_{NO}) \end{array} \tag{4.2a}$$

$$\sum_{v\in W_2} E_{i_1 v} \leq \sum_{v\in W_1} S_{i_2 v} + H\cdot(1-\sum_{v\in W_1} Z_{i_2 v}) \qquad \begin{array}{r} i_1,i_2\in T, i_1<i_2, \\ (W_1;W_2)\in biclique(G_{NO}) \end{array} \tag{4.2b}$$

Table 4.2: Cliques and bicliques selections a, b, and c for COSP2 and COSP3.

Cst	No.	Selection a	Selection b	Selection c
(i)	1	$\{1,2,3\}$	$\{1,2,3\}$	implied by 2, 3, 4
(ii)	2	$\{1,4\}$, $\{1,5\}$	$(\{1\};\{4,5\})$	$(\{1\};\{1,2,3,4,5\})$
	3	$\{2,6\}$, $\{2,7\}$, $\{2,8\}$	$(\{2\};\{6,7,8\})$	$(\{2\};\{1,2,3,6,7,8\})$
	4	$\{3,9\}$, $\{3,10\}$	$(\{3\};\{9,10\})$	$(\{3\};\{1,2,3,9,10\})$
	5	$\{4,11\}$, $\{6,11\}$	$(\{4,6\};\{11\})$	$(\{4,6,11,12\};\{11\})$
	6	$\{5,12,13\}$, $\{7,12,13\}$, $\{9,12,13\}$	$(\{5,7,9\};\{12,13\})$	$(\{5,7,9,12,13\};\{12,13\})$
	7	$\{8,14\}$, $\{10,14\}$	$(\{8,10\};\{14\})$	$(\{8,10,13,14\};\{14\})$
(iii)	8	implied by 6	$\{12,13\}$	implied by 6
(iv)	9	$\{11,12\}$	$\{11,12\}$	implied by 5
	10	$\{13,14\}$	$\{13,14\}$	implied by 7

There is a clear trade-off between the tightness of the LP relaxation and the size of the model. Therefore, it is important to carefully select the cliques and bicliques that are used to enforce non-overlapping constraints. We introduce 3 different selection heuristics. The first selection strategy, denoted by a, consists of selecting all maximal cliques only, leading to 15 constraints (4.1). The second selection strategy, denoted by b, consists of deriving cliques and bicliques directly from the non-overlapping constraint definitions. The third selection strategy, denoted by c, consists of improving selection b by extending it to maximal cliques and bicliques and removing unnecessary elements from the selection. Table 4.2 displays selections a, b and c. The first column indicates the descriptive constraint (as introduced in section 3.2.1) that generates the corresponding cliques or bicliques. For example, non-overlapping constraint (ii) for the first storage tank is intuitively represented by the clique $(\{1\};\{4,5\})$ in selection b but it is extended to the maximal biclique $(\{1\};\{1,2,3,4,5\})$ in selection c. Also, non-overlapping constraint (i) is represented by the maximal clique $\{1,2,3\}$ in selection b, but is removed in selection c as it is implied by bicliques 2, 3, and 4. Therefore, selection b leads to 16 non-overlapping constraints (4 of type (4.1) and 12 of type (4.2)) while selection c leads to 12 non-overlapping constraints (4.2) only.

4.3 Symmetry-Breaking Constraints

In section 3.3.7, a generic symmetry-breaking constraint was proposed in order to reduce the search space of the MOS model. Here, a problem-specific approach is developed in order to break the symmetries that arise in the SOS model. The methodology is based on the discrete representation of an SOS solution as a sequence of operations.

4.3.1 Symmetric Sequences of Operations

It is possible to determine different sequences of operations leading to the same schedule. For the optimal solution of COSP1 (Figure 3.3), one can show that there are 120 symmetric sequences of operations, 4 of which are displayed in Figure 4.1. These sequences are obtained from the optimal one by moving operations in the sequence in such a way that the same precedence constraints are active. As there are many different optimal discrete solutions, the branch & bound algorithm will explore many redundant nodes of the search tree. Therefore, an efficient symmetry-breaking tool is needed to avoid searching irrelevant solutions.

4.3.2 A Sequencing Rule Based on a Regular Language

A sequencing rule is defined in order to select the sequences of operations to be explored. This rule is expressed as a regular language, which can be recognized by a deterministic finite automaton (DFA). A regular language is a set of words (sequences of letters) defined from an alphabet (i.e. the set of operations), the empty word ε, and the operations concatenation '$\cdot$' (symbol usually omitted), union '$+$', and Kleene star '$*$'. Given two languages L_1 and L_2, these operations are defined by the following formulas.

$$L_1 \cdot L_2 = \{w = w_1 \cdot w_2 \text{ s.t. } w_1 \in L_1 \text{ and } w_2 \in L_2\}$$

$$L_1 + L_2 = \{w \text{ s.t. } w \in L_1 \text{ or } w \in L_2\}$$

$$L_1{}^* = \{\varepsilon\} \cup L_1 \cup L_1 \cdot L_1 \cup L_1 \cdot L_1 \cdot L_1 \cup \ldots$$

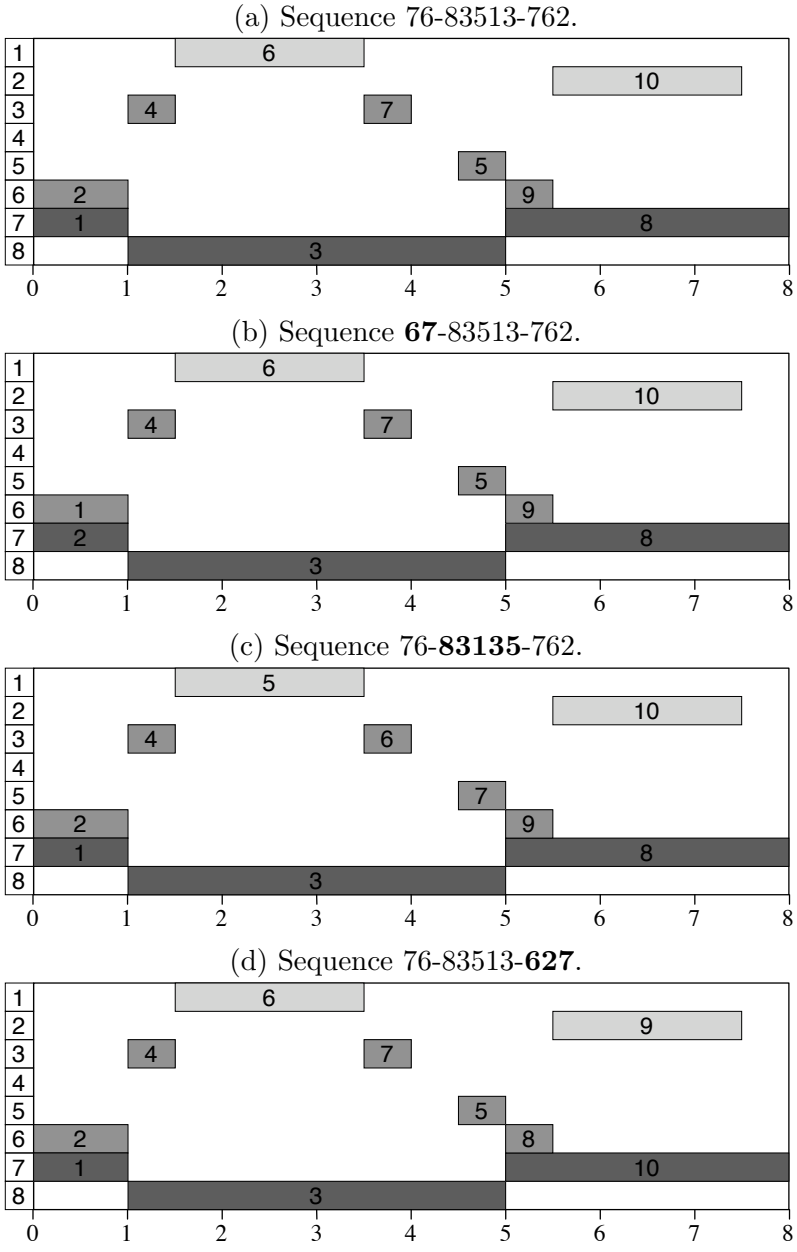


Figure 4.1: Symmetric sequences of operations for COSP1.

The reader may refer to Hopcroft and Ullman (1979) for a complete definition of regular languages.

4.3.3 Rule Derivation for COSP1

The rule presented in this section has been designed in order to remove as many symmetric sequences of operations as possible, regardless of its complexity. Notice that there is a trade-off between the symmetry-breaking capabilities of the sequencing rule (number of symmetric solutions it eliminates) and its complexity.

In COSP1, two distillation states exist as either distillation 7 or distillation 8 is being executed at any time. Thus, the sequence of operations can be decomposed into subsequences, each corresponding to one distillation state, as shown in Figure 4.1.

Let L_7 (resp. L_8) be the regular language describing the possible sequences of operations during distillation state 7 (resp. distillation state 8). Note that only transfer operations 1, 2, 4 and 6 are allowed to be executed during this state due to non-overlapping constraints.

If no unloading operation is performed, operations 4 and 6 need to be executed at most once. Thus, in that case, we choose to define the regular language L_7 so that a subsequence corresponding to the distillation state 7 starts with distillation 7 and may follow by at most one occurrence of transfer operations 4 and 6, in this order.

$$L_7 = \{7, 74, 76, 746\} = 7(\varepsilon + 4)(\varepsilon + 6)$$

If unloading operation 1 is allowed to be executed during distillation state 7, then it can be executed at most once. Also, it might be necessary to perform transfer operation 4 before and after this unloading. Thus, in that case, we define the regular language L_7 as follows.

$$L_7 = 7(\varepsilon + 4)(\varepsilon + 6)(\varepsilon + 1 + 14)$$

If both unloading operations 1 and 2 are allowed to be executed during distillation state 7, then it might also be necessary to perform transfer operation 6 before and after this unloading. Also, unloading 1 has to be sequenced before operation 2 due to the precedence constraint between unloading operations (crude-oil vessels have to unload in order of arrival). Thus, in this general case, we choose to define the regular language L_7 as follows.

$$L_7 = 7(\varepsilon + 4)(\varepsilon + 6)(\varepsilon + 1 + 14)(\varepsilon + 2 + 26)$$

Table 4.3: List of sequences belonging to regular language L_7.

Length	Sequences belonging to L_7
1	7
2	71, 72, 74, 76
3	712, 714, 726, 741, 742, 746, 761, 762
4	7126, 7142, 7412, 7414, 7426, 7461, 7462, 7612, 7614, 7626
5	71426, 74126, 74142, 74612, 74614, 74626, 76126, 76142
6	741426, 746126, 746142, 761426
7	7461426

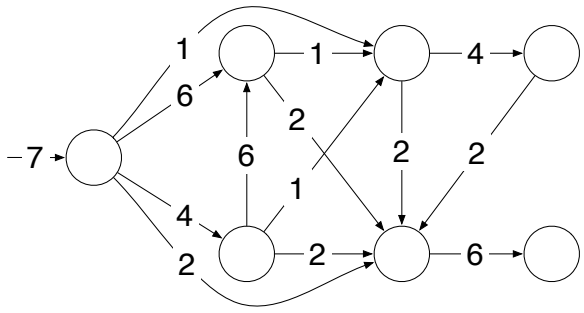


Figure 4.2: Automaton DFA_7 recognizing regular language L_7.

The set of all sequences of operations belonging to the regular language L_7 is displayed in Table 4.3. It can be represented by the DFA depicted in Figure 4.2, noted DFA_7. This DFA reads a sequence starting with operation 7, and then reads the following operation in the sequence by moving though the corresponding labeled arc. A sequence is accepted if it can be entirely read by DFA_7.

Similarly, the regular language L_8, represented by DFA_8, is defined as follows.

$$L_8 = 8(\varepsilon + 3)(\varepsilon + 5)(\varepsilon + 1 + 13)(\varepsilon + 2 + 25)$$

Finally, the regular language L describing an efficient sequencing rule for COSP1 can be defined using language L_7 and L_8. Figure 4.3 shows a scheme of the DFA recognizing the regular language L.

$$L = (\varepsilon + L_7)(L_8 \cdot L_7)^*(\varepsilon + L_8)$$

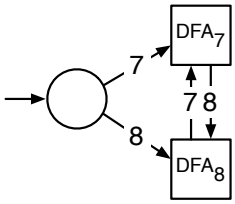


Figure 4.3: Automaton recognizing the regular language L.

It should be noted that this symmetry-breaking rule captures all possible schedules and removes many redundant sequences of operations. In Figure 4.1, schedule (a) is accepted by the sequencing rule while the other schedules are rejected, and thus not explored during the search. However, there are some symmetric sequences that remain accepted such as **78132** and **71832**. Indeed, in these two sequences belonging to the language L, exchanging operations 1 and 8 does not change the active precedence constraints.

4.3.4 Regular Constraint

Once the regular language L and its corresponding DFA have been defined, it is possible to include the sequencing rule using the linear system of equations proposed by Côté et al. (2007). The equations represent a unit network flow through a directed layered graph initially introduced by Pesant (2004) for constraint programming.

Let $M = (Q, \Sigma, \delta, q_1, F)$, where Q is the set of states (i.e. nodes), Σ is the alphabet, δ is the transition function, q_1 is the initial state, and F is the set of final states, be a DFA recognizing the regular language L. The idea is to unfold the automaton states on $n + 1$ layers (where n is the length of the sequence), the first layer corresponding to the initial state, the last layer corresponding to the possible final states, and the transition between layers corresponding to the automaton transitions defined by δ. Then, a sequence is recognized by the DFA M if and only if there is a flow of unit 1 from the initial state in the first layer to a final state in the last layer such that the transition taken between any two adjacent layers i and $i + 1$ corresponds to the ith letter of the sequence.

The regular language membership constraint makes use of flow binary variables S_{ivq},

where $q \in Q$. $S_{ivq} = 1$ if the automaton is in state q when operation v is assigned to priority-slot i, $S_{ivq} = 0$ otherwise. The network flow problem is defined with the following constraints.

$$\sum_{q} S_{ivq} = Z_{iv} \qquad i \in T, v \in W \qquad \text{linking constraint} \qquad (4.3a)$$

$$\sum_{v} S_{1vq_1} = 1 \qquad \text{initial unit flow} \qquad (4.3b)$$

$$\sum_{v,q',q=\delta(q',v)} S_{(i-1)vq'} - \sum_{v} S_{ivq} = 0 \qquad i \in T, i \neq 1, q \in Q \qquad \text{flow conservation} \qquad (4.3c)$$

$$\sum_{v,q,\delta(q,v)\in F} S_{nvq} = 1 \qquad \text{final unit flow} \qquad (4.3d)$$

Constraint (4.3a) links the Z_{iv} variable to the S_{ivq} variables ($q \in Q$) ensuring that the assignment of operation v to priority-slot i is equivalent to a unit flow traversing one of the corresponding arcs in the network. Constraint (4.3b) sets the flow leaving the network source node $(1, q_1)$ to 1. Constraint (4.3c) ensures that the flow entering network node (i, q) is equal to the flow leaving it. Constraint (4.3d) sets the flow leaving the network nodes (n, q) through transitions $\delta(q, v) \in F$ to 1.

Although $O(|T| \cdot |W| \cdot |Q|)$ new variables and $O(|T| \cdot |W| + |T| \cdot |Q|)$ new constraints are added to the model, the search space is substantially reduced as only sequences belonging to the regular language L are explored. It should be noted that it is not necessary to declare the variable S_{ivq} as binary. Indeed, if all variables Z_{iv} are fixed, then the sequence of operations is fixed to $v_1...v_i...v_n$. As the automaton M is deterministic, there is a unique sequence of states $q_1...q_i...q_nq_{n+1}$ (where $q_{n+1} \in F$) corresponding to the states visited upon processing the sequence $v_1...v_i...v_n$. Therefore, the network flow problem stated above has a unique solution defined by $S_{ivq} = 1$ if $v = v_i$ and $q = q_i$, $S_{ivq} = 0$ otherwise.

4.4 Computational Results

In this section, we present experimental results obtained on an Intel Xeon 1.86GHz processor with GAMS as modeling language, CPLEX 11 as MILP solver and CONOPT 3 as local

Table 4.4: SOS computational results for crude-oil operations scheduling problems.

Pb	n	LP	MILP	Nb of nodes	CPU
COSP1^a	13	80.000	79.750	18	5.88s
COSP1^b	13	80.000	79.750	19	4.45s
COSP1^c	13	80.000	79.750	21	4.92s
COSP2^a	21	103.000	101.175	36	120.42s
COSP2^b	21	103.000	101.175	19	55.57s
COSP2^c	21	103.000	101.175	25	60.50s
COSP3^a	21	100.000	87.400	28	191.47s
COSP3^b	21	100.000	87.400	33	97.70s
COSP3^c	21	100.000	87.400	31	64.46s
COSP4^a	26	132.585	132.548	16	606.86s
COSP4^b	26	132.585	132.548	37	574.95s
COSP4^c	26	132.585	132.548	32	308.43s

NLP solver. The DFAs for problems COSP2, COSP3, and COSP4 have been defined in a similar way as for COSP1, adapted to each refinery configuration.

4.4.1 Performance of the SOS Model

The MILP relaxation (defined in section 3.5.5) of the SOS model is solved using the direct approach (see section 2.6). The number of priority-slots is set to its maximum value as determined in section 4.4.2. Computational results are given in Table 4.4 for each clique/biclique selection heuristic. The behavior of this time representation is quite different than for the MOS and MOS-SST representations as COSP4 is now the most difficult while COSP2 is quite easy to solve. Indeed, COSP4 is solved with the largest number of priority-slots as it is the largest instance in terms of operations and resources, which makes it the hardest instance. The results obtained for selection a shows that using only maximal cliques is not the most efficient in the SOS model. It is rather preferable to combine it with bicliques as in selections b and c. Additionally, it is clear that the selection improvements in selection c lead to a significant decrease in CPU time for COSP3 and COSP4. This is due to the fact that non-overlapping constraints are strengthened and the model size is reduced.

4.4.2 Effect of the Number of Priority-Slots

It is critical to postulate an appropriate number of priority-slots to be used in the model. Indeed, a small number of priority-slots may lead to infeasibility, while a large number of priority-slots may result in an unsolvable model. The strategy used to determine the number of priority-slots is dependent on the problem. We introduce the concepts of minimum sequence and minimum schedule.

Definition 1 (minimum sequence). We denote NO_v the set of operations that must not overlap with operation $v \in W$. A *minimum sequence* $s = s_1 s_2 \dots s_n$ is a sequence of operations that satisfies the following property:

$$\forall i, j \in T, i < j, \forall v \in W, \quad (s_i = s_j = v) \Rightarrow (\exists k \in T, i < k < j, \exists w \in NO_v \setminus \{v\}, s_k = w)$$

Definition 2 (minimum schedule). A *minimum schedule* is a feasible schedule that can be generated by a minimum sequence s.

In other words, a minimum schedule is a schedule such that between any two executions of an operation v, another non-overlapping operation $w \in NO_v \setminus \{v\}$ is performed between these two executions. By merging any two executions of an operation v which do not respect the previous property, it can be shown that any optimal schedule for the crude-oil scheduling problem can be transformed into a minimum schedule. Therefore, it is sufficient to explore the space of minimum schedule. The symmetry-breaking rule presented earlier only accepts minimum sequences.

In the case of the refinery problems introduced by Lee et al. (1996), it can be shown that the number of operations in a minimum schedule is bounded as long as the number of distillation operations is bounded. We use the terminology *maximum number of operations* to represent the maximum length of a minimum sequence. For example, if COSP1 is constrained to be solved with a maximum of 3 distillation operations, the maximum number of operations is 13 as in the sequence **7461483525746**. Any sequence of 14 operations is not minimum, and thus will be rejected by the sequencing rule in combination with the distillation cardinality constraints. Therefore, it is unnecessary to postulate a number of

priority-slots greater than 13, whether the symmetry-breaking rule is used or not. The maximum number of operations is 21 for COSP2 and COSP3 (with at most 5 distillation operations), 26 for COSP4 (with at most 7 distillation operations).

Figure 4.4 shows the evolution of the number of nodes explored, the CPU time, the MILP and NLP solutions objective value ($100,000 unit), and the optimality gap with respect to the number of priority-slots when solving COSP1 to COSP4. Xpress was used to solve the MILP relaxation. The grey area represents the gap between MILP and NLP solutions. For all problems, the computational expense is small when the number of priority-slots is small, as the size of the problem and the feasible space are small. Also, the computational expense is small when the number of priority-slots is large (close to its maximum), as the solver must assign one operation to each priority-slot while satisfying both the cardinality constraints and the symmetry-breaking rule, thus reducing the size of the feasible space. In between, the number of nodes explored and the CPU time reach a maximum, although not at the same number of priority-slots. It can also be observed that the objective value of the optimal solutions of the MILP relaxation increases with the number of priority-slots as more flexibility is given to the solver to find feasible solution. The optimality gap tends to decrease.

4.4.3 Remark on the Optimality of the Solution

When no symmetry-breaking rule is used, the global optimal solution of the SOS model with maximum number of priority-slots leads to the best possible schedule. Indeed, the optimal sequence of operations can be extended with unused operations, for which the volume transferred is set to 0. Thus, it can be obtained even if the user postulated more priority-slots than needed. However, in general, if a sequencing rule is used, the global optimal solution might not be obtained with maximum number of priority-slots, but it will be obtained with a number of slots identical to the number of operations actually performed in this solution. For example, consider the sequence **71287** and assume it is optimal. In this case, unloading operations 1 and 2 may each overlap with two consecutive distillations.

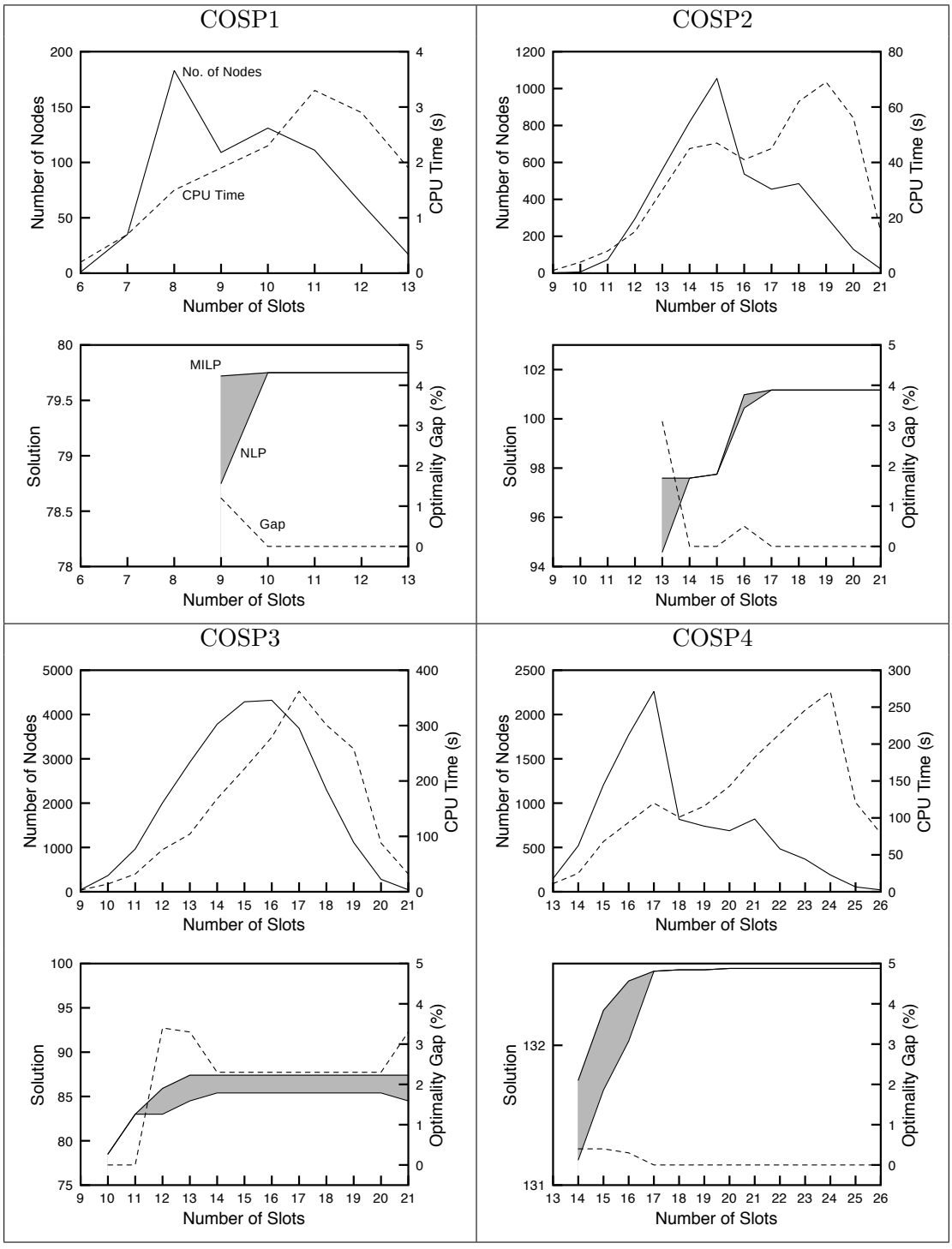


Figure 4.4: Performance of the SOS model on crude-oil scheduling problems (MILP solver: Xpress).

However, if the model is solved with 13 slots using the sequencing rule, unloading operations cannot overlap with two consecutive distillations. Indeed, if unloading 1 is sequenced after a distillation 7 and before a distillation 8, transfer 4 will necessarily be sequenced before and after unloading 1, within the current subsequence for distillation state 7 (i.e. **74614** or **7461426**). Therefore, due to precedence constraints, unloading 1 is forced to be executed entirely during distillation 7, and cannot overlap with next or previous distillations. Nonetheless, in our experiments, the best solutions were always obtained with the maximum number of priority-slots which tends to justify this choice for the parameter n.

4.4.4 Effect of Symmetry-Breaking Constraints

In this section, we study the effect of the symmetry-breaking rule on solving the MILP relaxation of the SOS model, which is the first step of the MILP-NLP decomposition procedure presented in section 3.5.5. Two models will be compared: the basic model that includes all linear constraints of the MILP except the symmetry-breaking constraints, and the extended model that includes all linear constraints as well as the symmetry-breaking constraints.

Consider a single instance of COSP1 (Figure B.1, Table B.1) for which 13 priority-slots are postulated. As shown in Table 4.5, although the extended model contains many more new variables and relatively fewer additional constraints than the basic model, the number of nodes explored is greatly reduced (from more than one million to 63), which leads to very low CPU time (from more than 3600s to 2s). This is due to the removal of many symmetric solutions from the search space by using the symmetry-breaking rule. It should be noted that both models have the same LP relaxation. The optimal solution of the basic model is found early during the search, but a large amount of time is used to prove its optimality.

Figure 4.5 shows how both models perform on COSP1 when varying the number of priority-slots from 6 to 13. Clearly, the computational expense needed to solve the basic model grows exponentially with the number of priority-slots. However, both the number of nodes and the CPU time remain stable when solving the extended model.

Table 4.5: Size and performance of Basic and Extended models on COSP1 (13 slots).

	Variables	Binary Variables	Constraints	MILP	Nodes	CPU Time
Basic	1,392	104	2,515	79.75	+180,000	+3,600s
Extended	2,801	104	2,952	79.75	21	5s

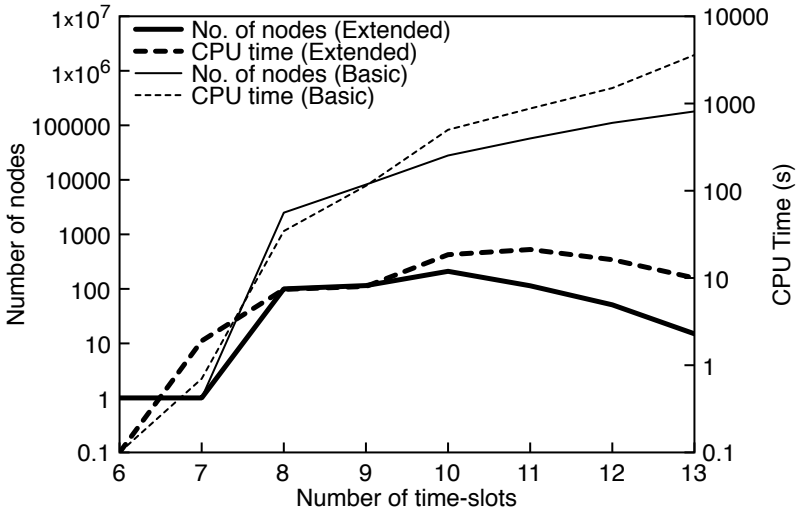


Figure 4.5: Performance of the Basic and Extended models on COSP1 (6 to 13 slots).

4.5 Comparison of Crude-Oil Scheduling Models

In this section, we compare the four time representations applied to the crude-oil operations scheduling problem in chapter 3 and 4. Figure 4.6 shows how the objective value of the best incumbent varies over time depending on the time representation used. Results for the SOS model are presented for clique/biclique selection heuristic *c*. As for the batch scheduling problems the MOS model proves to be superior. However, the MOS-SST model behaves better than the MOS-FST model for these problems due to non-constant processing times. The SOS model performs better than the MOS-SST model for instance COSP2, worse for instance COSP4, and similarly for the other instances. From these results, it seems that the MOS-SST model scales better with the size of the problem than the SOS model as it requires fewer priority-slots. Finally, in Table 4.6 we present the model sizes of the MILPs

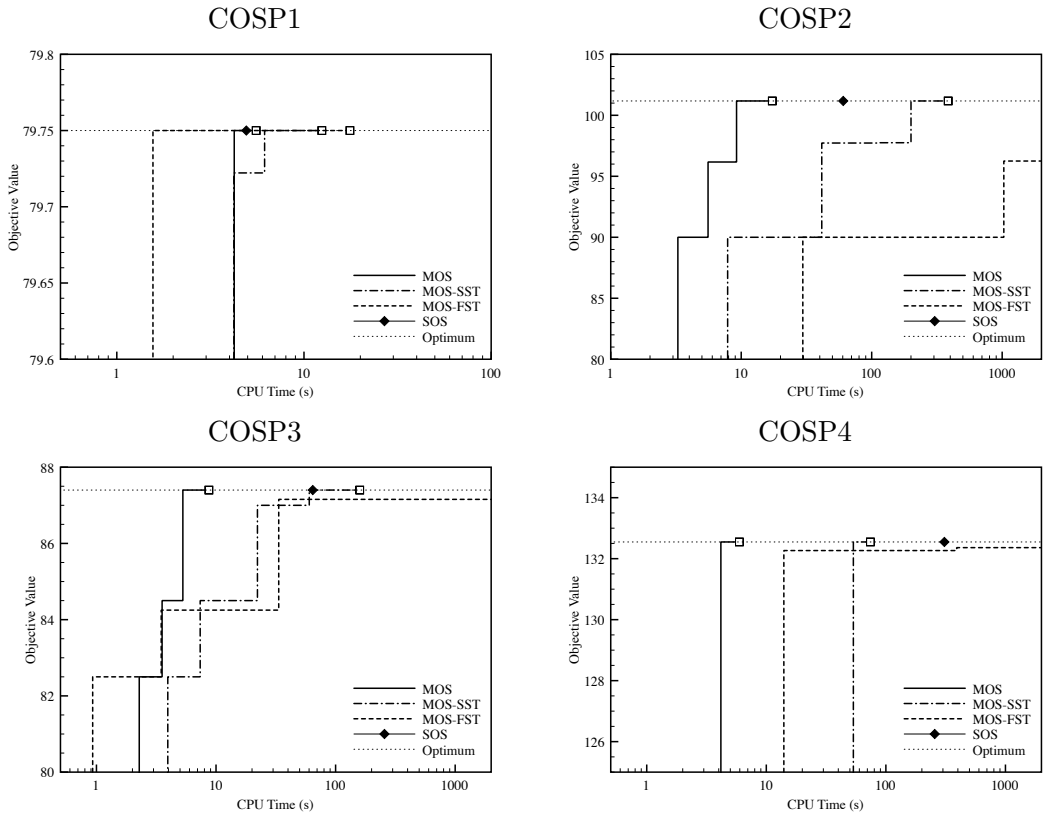


Figure 4.6: Comparison of time representations for crude-oil scheduling problems.

corresponding to all time representations using the number of priority-slots leading to the best solution. It is clear that the MOS time representation leads to the most compact models which partly explains its efficiency. The MOS-FST time representation is much larger due to the higher number of priority-slots required. Furthermore, it is interesting to note that the SOS time representation requires many more continuous variables. These variables are introduced to represent the network flow used to break symmetries as described in section 4.3. Overall, it is also interesting to note that all four models provide the same LP relaxation for each instance. Indeed, the objective function is purely economic and not directly linked to scheduling issues, as opposed to the previous batch scheduling examples. In the case of crude-oil operations scheduling, solving the LP relaxation corresponds to generating a solution that only satisfies overall material balances, distillation demands and

Table 4.6: Size of MOS, MOS-SST, MOS-FST, and SOS models for crude-oil scheduling problems.

Problem	Model	n	Binary Vars	Continuous Vars	Constraints
COSP1	MOS	5	40	496	1,086
	MOS-SST	6	48	601	1,403
	MOS-FST	8	64	793	1,804
	SOS	13	104	2,801	2,952
COSP2	MOS	6	84	1,303	2,620
	MOS-SST	8	112	1,745	3,758
	MOS-FST	16	224	3,473	8,051
	SOS	21	294	13,084	7,835
COSP3	MOS	5	70	1,211	2,302
	MOS-SST	7	98	1,702	3,431
	MOS-FST	16	224	3,873	8,439
	SOS	21	294	13,609	7,909
COSP4	MOS	4	76	1,489	2,806
	MOS-SST	6	114	2,239	4,338
	MOS-FST	16	304	5,953	12,269
	SOS	26	494	28,445	14,351

specifications, and capacity limits at the end of the scheduling horizon.

4.6 Conclusion

In this chapter, the SOS time representation has been applied to crude-oil scheduling problems with the objective of maximizing gross margins. The corresponding mathematical model is strengthened using the non-overlapping graph of the scheduling system. It is possible to represent a solution schedule as a single sequence of operations that can be restricted with respect to a problem-specific DFA-based symmetry-breaking sequencing rule. Computational results show that using symmetry-breaking constraints is critical in order to solve the SOS model in reasonable times. Similarly to the batch scheduling problems studied in chapter 2, the MOS time representation is the most efficient approach. The SOS model compares well with the MOS-SST model, while the discrete-time MOS-FST time representation is clearly not suitable as the problem involves variable processing times.

Chapter 5
Tightening the Linear Relaxation of a Crude-Oil Operations Scheduling MINLP Using Constraint Programming

5.1 Introduction

In this chapter, we consider the optimization of the crude-oil operations scheduling problem with the objective of minimizing the total logistics costs, as studied in Lee et al. (1996). Logistics costs include sea waiting and unloading costs for marine vessels, storage costs in tanks, and CDU switching costs. In earlier work, an operation specific event point continuous-time formulation has been developed (see Jia et al., 2003) and applied to the problems from Lee et al. (1996) (COSP1, ..., COSP4) using a linear approximation of storage costs. However, no proof of optimality or estimate of optimality gap is given with this approximation. The global optimization of this model was later addressed by Karuppiah et al. (2008) using an outer-approximation algorithm where the MILP master problem is solved by a Lagrangian decomposition. While rigorous, this method is still computationally expensive.

In chapter 3, we presented a solution procedure for solving non-convex crude-oil scheduling MINLPs that returns a suboptimal solution with an estimate of the gap with respect to the optimal solution. The main contribution of this chapter is to reduce the optimality gap by tightening the linear relaxation of the MINLP using McCormick convex envelopes for bilinear products of continuous variables (see McCormick, 1976). This can be done by using discretization techniques and applying McCormick convex and concave estimators on

each discrete element of the continuous space (see Bergamini et al., 2008; Wicaksono and Karimi, 2008; Pham et al., 2009). On the other hand, it has been shown how Constraint Programming (CP) techniques can be effectively integrated with branch & bound procedures for the global optimization of MINLPs (see Sahinidis, 2003). In this chapter, CP is used during the search to tighten variable bounds and generate new McCormick cuts, thus effectively tightening the mixed-integer linear program (MILP) relaxation. When applied to the sequential MILP-NLP approach, this extension of the branch & cut algorithm leads to reduced optimality gaps and allows finding better suboptimal solutions with lower logistics cost. The concepts developed in this work can be applied to a generic solver as in the global MINLP solver BARON, or can be used to effectively solve other optimization problems to global optimality by exploiting their specific structure.

This chapter is organized as follows. First we present the MINLP mathematical formulation with the nonlinear objective function. The model is then reformulated in order to get a linear objective function and obtain a simple MILP relaxation using McCormick convex envelopes. Next, this MILP relaxation is improved by adding tighter McCormick cuts for some subproblems explored during the branch & cut procedure. Finally, we provide computational results showing the impact of this approach in terms of relaxation tightness and CPU time.

5.2 MINLP Model

In this section, the SOS time representation is used to derive the MINLP model for the crude-oil scheduling problem. It differs from the SOS model presented in appendix C by a different objective function and an additional constraint that defines the sea waiting time for each crude vessel.

The objective function minimizing total logistics costs can be expressed as follows:

$$\begin{aligned}\min \quad & SWITCHINGCOST \sum_{r \in R_D} \left(\sum_{i,v \in I_r} Z_{iv} - 1 \right) \\ & + UNLOADINGCOST \sum_{i \in T} \sum_{v \in W_U} D_{iv} \\ & + SEAWAITINGCOST \sum_{i \in T} \sum_{v \in W_U} W_{iv} \\ & + FIXEDSTORAGECOST \\ & + \sum_{i \in T} \sum_{r_1, r_2 \in R, v \in O_{r_1} \cap I_{r_2}} (SC_{r_2} - SC_{r_1}) \left(H - S_{iv} - \frac{D_{iv}}{2} \right) \cdot V_{iv}\end{aligned}$$

where:

- W_{iv} is the variable waiting time before unloading v if it is assigned to slot i, $W_{iv} = 0$ otherwise
- $SWITCHINGCOST$ is the cost associated with each CDU switch
- $UNLOADINGCOST$ is the unloading cost rate
- $SEAWAITINGCOST$ is the sea waiting cost rate
- $FIXEDSTORAGECOST$ is the total cost for storing the initial refinery inventory during the horizon H if no crude transfer is performed
- SC_r is the storage cost rate in resource r

The last term, which involves bilinearities making the model nonconvex, evaluates the variation of the total storage cost. For each transfer operation v between resources r_1 (storage cost rate SC_1) and r_2 (storage cost rate SC_2) assigned to priority-slot i, the corresponding variation of storage cost is calculated as follows:

$$\begin{aligned}\Delta COST_{iv} &= (SC_{r_2} - SC_{r_1}) \left(\int_{t=S_{iv}}^{t=S_{iv}+D_{iv}} \frac{V_{iv}}{D_{iv}} \cdot (t - S_{iv}) dt + \int_{t=S_{iv}+D_{iv}}^{t=H} V_{iv} dt \right) \\ &= (SC_{r_2} - SC_{r_1}) \left(\frac{1}{2} D_{iv} \cdot V_{iv} + (H - S_{iv} - D_{iv}) \cdot V_{iv} \right) \\ &= (SC_{r_2} - SC_{r_1}) \left(H - S_{iv} - \frac{D_{iv}}{2} \right) \cdot V_{iv}\end{aligned}$$

The variable W_{iv} is defined by the following constraint:

$$W_{iv} = S_{iv} - \underline{S_v} \cdot Z_{iv} \qquad i \in T, v \in W_U \tag{5.1}$$

The full model is written as follows:

$$\begin{aligned}
\min \quad & SWITCHINGCOST \sum_{r \in R_D} \left(\sum_{i,v \in I_r} Z_{iv} - 1 \right) \\
& + UNLOADINGCOST \sum_{i \in T} \sum_{v \in W_U} D_{iv} \\
& + SEAWAITINGCOST \sum_{i \in T} \sum_{v \in W_U} W_{iv} \\
& + FIXEDSTORAGECOST \\
& + \sum_{i \in T} \sum_{r_1, r_2 \in R, v \in O_{r_1} \cap I_{r_2}} (SC_{r_2} - SC_{r_1}) \left(H - S_{iv} - \frac{D_{iv}}{2} \right) \cdot V_{iv} \\
\text{s.t.} \quad & W_{iv} = S_{iv} - \underline{S_v} \cdot Z_{iv} \qquad i \in T, v \in W_U \\
& \text{Constraints (3.1)-(3.6), (3.8)-(3.10), (4.1)-(4.3), and (2.27)}
\end{aligned}$$

Table 5.1 displays the corresponding cost data for problems COSP1, ..., COSP4. The fixed storage cost is calculated as $\sum_r SC_r \cdot L^t_{0r}$. The additional parameters are given in appendix B. Note that constraint (3.6) contains the nonlinear composition constraints discussed in chapter 3. The model is intended to be solved with the MILP-NLP decomposition procedure introduced in section 3.5.5. In order to obtain near-optimal solutions, it is critical to derive a tight linear relaxation of the full MINLP. In particular, the objective function needs to be reformulated as explained in the following section. The nonlinear constraints (3.6c) are simply dropped in the MILP, as justified by the results obtain in chapter 3 in terms of optimality gap. However, these constraints are included in the second stage NLP.

Table 5.1: Cost data for crude-oil operations scheduling problems.

	COSP1	COSP2	COSP3	COSP4
Switching cost	50	50	50	30
Unloading cost	8	8	10	7
Sea waiting cost	5	5	5	5
Storage cost in storage tanks	0.05	0.05	0.04	0.05
Storage cost in charging tanks	0.08	0.08	0.08	0.06
Fixed storage cost	104	158	134.4	273

5.3 Reformulation and Linear Relaxation

It is common to reformulate MINLP problems by introducing new variables to represent nonlinear terms. The MINLP model can then be rewritten as follows:

$$
\begin{aligned}
\min \quad & SWITCHINGCOST \sum_{r \in R_D} \left(\sum_{i,v \in I_r} Z_{iv} - 1 \right) \\
& + UNLOADINGCOST \sum_{i \in T} \sum v \in W_U D_{iv} \\
& + SEAWAITINGCOST \sum_{i \in T} \sum v \in W_U W_{iv} \\
& + FIXEDSTORAGECOST \\
& + \sum_{i,r_1,r_2,v \in O_{r_1} \cap I_{r_2}} (SC_{r_2} - SC_{r_1})\, X_{iv} \\
\text{s.t.} \quad & X_{iv} = A_{iv} V_{iv} && i \in T, w \in W \\
& A_{iv} = H - S_{iv} - \frac{D_{iv}}{2} && i \in T, w \in W \\
& W_{iv} = S_{iv} - \underline{S_v} \cdot Z_{iv} && i \in T, v \in W_U \\
& \text{Constraints (3.1)-(3.6), (3.8), (3.10), (4.1)-(4.3), and (2.27)}
\end{aligned}
$$

where variables X_{iv} are used to represent the bilinear terms $A_{iv} \cdot V_{iv}$. Using McCormick convex envelopes introduced in McCormick (1976), this MINLP can be linearly relaxed by

replacing the constraint $X_{iv} = A_{iv} \cdot V_{iv}$ by:

$$X_{iv} \geq A_{iv}^L V_{iv} + A_{iv} V_{iv}^L - A_{iv}^L V_{iv}^L$$
$$X_{iv} \geq A_{iv}^U V_{iv} + A_{iv} V_{iv}^U - A_{iv}^U V_{iv}^U$$
$$X_{iv} \leq A_{iv}^L V_{iv} + A_{iv} V_{iv}^U - A_{iv}^L V_{iv}^U$$
$$X_{iv} \leq A_{iv}^U V_{iv} + A_{iv} V_{iv}^L - A_{iv}^U V_{iv}^L$$

The terms A_{iv}^L, A_{iv}^U, V_{iv}^L, and V_{iv}^U represent the lower and upper bounds of variables A_{iv} and V_{iv}. It is important to note that the tightness of this linear relaxation strongly depends on the bounds of the variables A_{iv} and V_{iv}.

5.4 McCormick Cuts

Once the linear relaxation of the MINLP has been defined, the corresponding MILP can be solved by a branch & cut algorithm as implemented in the commercial tool CPLEX. This approach consists of successively solving many subproblems (nodes) of the original MILP by solving its continuous relaxation and generating new subproblems when needed. User-defined constraints can also be added to each subproblem during the search using callbacks (see ILOG Inc., 2007).

A subproblem of the MILP is obtained by fixing some of the discrete variables Z_{iv} to either 0 or 1. Each of these subproblems correspond to a linear relaxation of the MINLP subproblem obtained by fixing the same discrete variables to the same values. If the LP relaxation of an MILP subproblem is integer feasible (i.e. all discrete variables take an integer value, whether it is fixed or not), it might still not satisfy all MINLP constraints. In such cases, it is possible to infer stronger McCormick constraints by contracting variables bounds for the current discrete solution.

We denote p a discrete solution defined by the discrete values taken by the variables Z_{iv}. The discrete solution p corresponds to a unique sequence of operations $(v_1^p, \ldots, v_n^p)$. We

denote $(CP)^p$ the following CP model:

$$(CP)^p \begin{cases} \text{MINLP constraints} \\ Z_{iv} = 1 \quad \forall (i,v), v = v_i^p \\ Z_{iv} = 0 \quad \forall (i,v), v \neq v_i^p \end{cases}$$

When a discrete solution p is obtained at a given node of the search tree, ILOG CP is used to perform constraint propagation on the model $(CP)^p$ leading to variable bounds $\left(A_{iv}^L\right)^p$, $\left(A_{iv}^U\right)^p$, $\left(V_{iv}^L\right)^p$, and $\left(V_{iv}^U\right)^p$ that are possibly tighter than the bounds defined at the root node (modeling stage). The McCormick constraints derived from these bounds are valid for the discrete solution p but are not valid for the current node as some variables Z_{iv} might not have been fixed yet. The NLP subproblem corresponding to the discrete solution p is in fact a *restriction* of the NLP subproblem corresponding to the current node. Therefore, the following modified big-M McCormick constraints are defined to be added to the current node subproblem:

$$X_{iv} \geq \left(A_{iv}^L\right)^p V_{iv} + A_{iv}\left(V_{iv}^L\right)^p - \left(A_{iv}^L\right)^p \left(V_{iv}^L\right)^p - M_1 \cdot (n - \sum_i Z_{iv_i^p})$$

$$X_{iv} \geq \left(A_{iv}^U\right)^p V_{iv} + A_{iv}\left(V_{iv}^U\right)^p - \left(A_{iv}^U\right)^p \left(V_{iv}^U\right)^p - M_2 \cdot (n - \sum_i Z_{iv_i^p})$$

$$X_{iv} \leq \left(A_{iv}^L\right)^p V_{iv} + A_{iv}\left(V_{iv}^U\right)^p - \left(A_{iv}^L\right)^p \left(V_{iv}^U\right)^p + M_3 \cdot (n - \sum_i Z_{iv_i^p})$$

$$X_{iv} \leq \left(A_{iv}^U\right)^p V_{iv} + A_{iv}\left(V_{iv}^L\right)^p - \left(A_{iv}^U\right)^p \left(V_{iv}^L\right)^p + M_4 \cdot (n - \sum_i Z_{iv_i^p})$$

where:

$$M_1 = \left(A_{iv}^L\right)^p V_{iv}^U + A_{iv}^U\left(V_{iv}^L\right)^p - \left(A_{iv}^L\right)^p \left(V_{iv}^L\right)^p$$

$$M_2 = \left(A_{iv}^U\right)^p V_{iv}^U + A_{iv}^U\left(V_{iv}^U\right)^p - \left(A_{iv}^U\right)^p \left(V_{iv}^U\right)^p$$

$$M_3 = A_{iv}^U V_{iv}^U - \left\{\left(A_{iv}^L\right)^p V_{iv}^L + A_{iv}^L\left(V_{iv}^U\right)^p - \left(A_{iv}^L\right)^p \left(V_{iv}^U\right)^p\right\}$$

$$M_4 = A_{iv}^U V_{iv}^U - \left\{\left(A_{iv}^U\right)^p V_{iv}^L + A_{iv}^L\left(V_{iv}^L\right)^p - \left(A_{iv}^U\right)^p \left(V_{iv}^L\right)^p\right\}$$

A lazy constraint callback (see ILOG Inc., 2007) has been implemented in order to generate these local McCormick cuts at each node where an integer feasible solution is found.

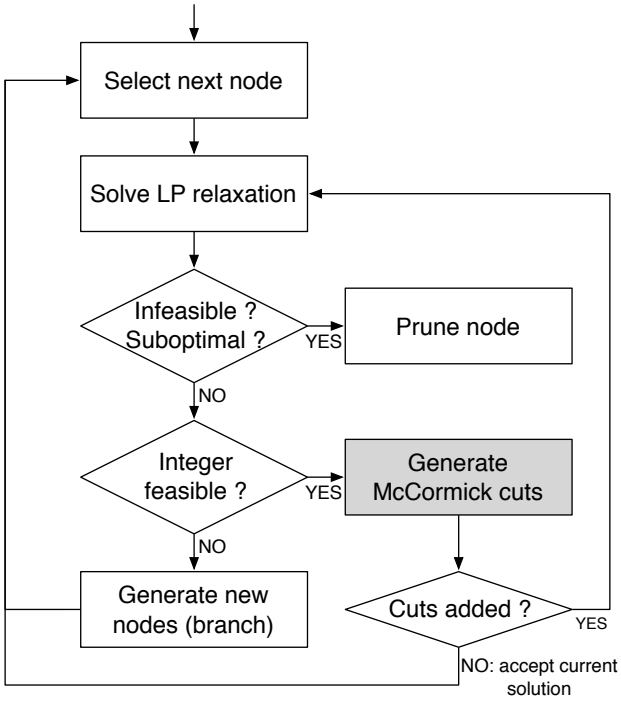


Figure 5.1: Branch & cut algorithm with McCormick cuts.

If at least one McCormick cut is violated by the current discrete solution, it is added to the node subproblem and the LP relaxation is resolved as depicted in Fig. 5.1. This cut generation procedure can also be executed at nodes where no integer feasible solution is found. However, in order to reduce the computational expense corresponding to the generation of variable bounds and cuts, only integer feasible nodes are processed.

5.5 Computational Results

The four problems introduced in Lee et al. (1996) have been solved using two algorithms. The *BasicRelaxation* algorithm consists of initially adding McCormick constraints to the MILP model without generating new cuts during the search. The *ExtendedRelaxation* algorithm consists of adding McCormick constraints to the MILP and new cuts during the search, as explained in the previous section. Both approaches have been developed in

C++ using CPLEX 11 (MILP) and ILOG CP 6.5 (CP). The NLPs have been solved using CONOPT 3. Experiments have been run on an Intel Core 2 Duo 2.16GHz processor.

In both basic and extended cases, Table 5.2 gives the solution of the MILP and the NLP, the corresponding optimality gap, and the total CPU time and the number of nodes explored during the search. In the ExtendedRelaxation case, the computational time for generating McCormick cuts using CP is also displayed in column "CP", it is already included in the total CPU time. The computational time for solving NLPs is not reported as it is always lower than 5s.

The results show that using the approach developed in this paper leads to important reductions of the optimality gap compared to the *BasicRelaxation* algorithm (3.48% vs 14.83% average gap). Besides, a better feasible solution (3.3% cost reduction) has been found for COSP2 using the two-step MILP-NLP procedure. This demonstrates how the linear relaxation of the MINLP can been tightened by adding McCormick constraints to the subproblems of the MILP leading to integer feasible solutions. More precisely, at the node where the optimal MILP solution of COSP1 is found, two rounds of cuts are added. The first round of cuts leads to an increase of the objective value from 199.1 to 212.4 (6.7% increase), while the second round increases the objective value to 213.7 (0.6% increase). This indicates that few rounds of cuts are necessary in this case and that the first round of cuts is the most important in order to tighten the MILP relaxation.

The optimality gaps obtained in the *BasicRelaxation* case are much larger than the optimality gaps obtained in chapters 3 and 4. This is due to the fact that the objective function considered in this previous work is linear, thus leading to a tighter MILP relaxation.

In terms of computational expense, the required CPU time increases by 9.5% for the *ExtendedRelaxation* procedure. This is due to the increase of the number of nodes explored in some cases (COSP2 and COSP3), the increase of the model size for subproblems for which McCormick constraints have been generated, and the time used for performing constraint propagation on the CP model (4.7% of total CPU time). This last point can be improved with a better CP model and a more efficient implementation.

Table 5.2: Results obtained with BasicRelaxation and ExtendedRelaxation algorithms.

BasicRelaxation					
Pb	MILP			NLP	Gap
	Solution	CPU	Nodes		
COSP1	199.1	9s	22	222.3	11.7%
COSP2	297.8	215s	55	**362.9**	21.9%
COSP3	254.6	224s	73	287.6	13.0%
COSP4	331.8	600s	20	374.0	12.7%

ExtendedRelaxation						
Pb	MILP				NLP	Gap
	Solution	CPU	CP	Nodes		
COSP1	213.7	11s	1s	20	222.3	4.0%
COSP2	343.1	246s	19s	57	**351.2**	2.4%
COSP3	269.2	337s	16s	95	287.6	6.8%
COSP4	371.3	554s	18s	19	374.0	0.7%

Table 5.3 shows computational results obtained with different MINLP algorithms on COSP1. Local optimizers DICOPT, SBB, Bonmin and AlphaECP have been tested as well as the global solver BARON. DICOPT and SBB did not return the best known solution, SBB failed to find any feasible solution. However, the corresponding CPU times are similar to the BasicRelaxation and ExtendedRelaxation procedures. Bonmin-OA found the best-known solution in reasonable time. AlphaECP and BARON also found the best known solution, but the former requires one order of magnitude increase in CPU time and does not give any optimality gap estimate, while the latter requires more than two orders of magnitude increase in CPU time (optimization is stopped after 1 hour) but has the smallest optimality gap.

5.6 Conclusion

In this chapter, a new approach for handling bilinear terms in MINLPs has been presented. It involves using CP bound contraction techniques during the branch & cut search in order to generate cuts based on McCormick convex envelopes for products of continuous variables. The procedure has the advantage of using the complementary strengths for CPLEX and

Table 5.3: Results obtained with diferrent MINLP algorithms on COSP1 and COSP2.

	COSP1			COSP2		
Algorithm	Solution	CPU	Gap	Solution	CPU	Gap
BasicRelaxation	222.3	9s	11.7%	362.9	215s	21.9%
ExtendedRelaxation	222.3	11s	4.0%	351.2	246s	2.4%
DICOPT	233.5	14s	-	351.2	1235s	-
SBB	Local infeas.	14s	-	Local infeas.	697s	-
Bonmin-OA	222.3	27s	-	No solution	+3,600s	-
AlphaECP	222.3	260s	-	358.0	+3,600s	-
BARON	222.3	+3,600s	4.1%	No solution	+3,600s	-

ILOG CP to obtain better solutions and reduce the optimality gap. Due to the tight integration of both tools using the modeling technology ILOG CONCERT, the increase in solution times remains small. Although the proposed approach has been applied to a crude-oil scheduling SOS MINLP model, it can also be applied to other time representations, such as MOS, with no major modifications.

This approach may be improved by extending the interaction between MILP and CP as in the programming system SCIP (see Achterberg, 2004) or in the integrated solver SIMPL (see Yunes et al., 2010), although a conceptual difference lies in the fact that CPLEX is used as the main mixed-integer branch & cut algorithm. Also, MINLP cuts such as McCormick cuts can be generated not only for integer feasible subproblems, but for other subproblems, thus pruning additional nodes during the search. Finally, optimality-based reduction techniques (see Sahinidis, 2003) can be used to add new constraints to the CP model to remove feasible solutions that are not optimal. As a consequence variable bounds may be further contracted leading to a tighter MILP relaxation.

Chapter 6
Integration of Refinery Planning and Crude-Oil Scheduling using Lagrangian Decomposition

6.1 Introduction

Although, integration of planning and scheduling has recently been addressed in the context of multiproduct continuous and batch production plants (see Erdirik-Dogan and Grossmann, 2008; Maravelias and Sung, 2009), very little work has been done towards the integration of planning and crude-oil scheduling problems in the context of refineries. This is due to the fact that in this case, the planning model is not an aggregate scheduling model. Therefore, the decomposition methods developed for batch and continuous plants are not directly applicable to refineries. In particular, planning and scheduling correspond to two different problems solely linked through CDU feedstocks. Therefore, instead of using a hierarchical decomposition, a spatial Lagrangian decomposition is preferred. The reader may refer to Fisher (1985) and Guignard (2003) for extensive reviews on Lagrangian relaxation and decomposition techniques. These approches have been applied to many industrial problems such as production planning and scheduling integration (see Li and Ierapetritou, 2010) or multiperiod refinery planning (see Neiro and Pinto, 2006). Thus, it seems natural to apply Lagrangian decomposition to solve the integrated refinery planning and crude-oil scheduling problem.

The content of this chapter is organized as follows. First, the planning and scheduling problems are stated as well as the full-space integrated problem. Next, a Lagrangian decom-

position scheme based on the dualization of CDU feedstock linking constraints is presented and a new hybrid method is introduced to solve the Lagrangian relaxation. A heuristic algorithm is then developed to obtain good feasible solutions for the integrated full-space problem. After some technical and practical remarks, we present numerical illustrations of the proposed approach on a small case study and a larger refinery problem.

6.2 Problem Statement

6.2.1 Refinery Planning Problem

The refinery planning problem can be regarded as a flowsheet optimization problem with multiple periods during which the refinery system is assumed to operate in steady-state. Due to extensive stream mixing, the model for each period is based on a pooling problem that is extended in order to include process models for each refining unit. The different periods in the model are connected through many material inventories. In this work, we consider a single-period planning model based on a pooling problem inspired from the literature (see for instance Adhya et al., 1999). A basic refinery planning system is represented in Figure 6.1. A set of crudes $i \in I$ are to be mixed in different types of crude-oil blends $j \in J$ (e.g. low-sulfur and high-sulfur blends), each associated to a specific CDU operating mode. For each mode and each crude, several distillation cuts are obtained with different yields. These crude cuts are then blended into intermediate pools which are used to prepare several final products. Therefore, the refinery planning system is composed of the following elements:

- One input stream for each selected crude $i \in I$ and each type of crude blend $j \in J$
- One CDU with fixed yields for each distillation cut
- Set of distillation cuts $k \in K$
- One pool for each type of crude blend $j \in J$ and each cut $k \in K$
- One intermediate stream between each pool $(j, k) \in J \times K$ and each final product $l \in L$

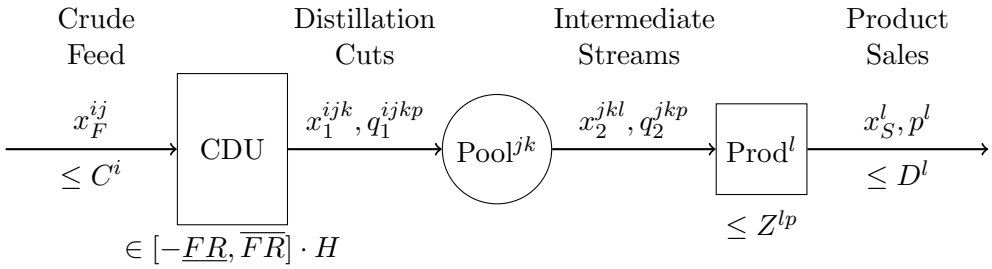


Figure 6.1: Basic refinery planning system.

- Set of final products $l \in L$
- One sales stream for each final product $l \in L$
- Set of qualities $p \in P$

The yield of crude $i \in I$ in distillation cut $k \in K$ when processed in crude blend $j \in J$ is assumed to be fixed and is denoted by α^{ijk}. In terms of stream qualities, it is assumed that distillation cuts have fixed qualities while pool qualities are calculated by bilinear quality balance constraints. A pure flow-based model is used to formulate the pooling problem as shown below. CDU flowrate limitations are considered independent of the operating mode and are enforced globally for all crudes processed during the period. The nomenclature used is as follows:

- Variables:
  - x_F^{ij} is a variable representing the amount of crude i selected for CDU distillation in blend j
  - x_1^{ijk} is a variable representing the amount of cut k extracted from crude i in blend j
  - x_2^{jkl} is a variable representing the flow of material between pool (j,k) and product l
  - q_2^{jkp} is a variable representing the quality p of pool (j,k)
  - x_S^l is a variable representing the amount of final product l sold

- Parameters:
  - q_1^{ijkp} is the (fixed) quality p of cut k extracted from crude i in blend j
  - p is the market value of final products
  - C^i is the amount of available crude i
  - α^{ijk} is the yield of cut k extracted from crude i in blend j
  - H is the planning horizon
  - $[\underline{FR}, \overline{FR}]$ is the bounds on CDU flowrate
  - D^l is the maximum demand in product l
  - Z^{lp} is the maximum specification for property p of product l

The NLP model is expressed as follows:

$$\max \quad \sum_{l \in L} p^l x_S^l \qquad \text{(sales revenue maximization)}$$

$$\text{s.t.} \quad 0 \leq \sum_{j \in J} x_F^{ij} \leq C^i \qquad i \in I \qquad \text{(crude availability)}$$

$$\underline{FR} \cdot H \leq \sum_{i \in I} \sum_{j \in J} x_F^{ij} \leq \overline{FR} \cdot H \qquad \text{(CDU flowrate limitations)}$$

$$x_1^{ijk} = \alpha^{ijk} \cdot x_F^{ij} \qquad (i,j,k) \in I \times J \times K \qquad \text{(CDU yield calculation)}$$

$$\sum_{i \in I} x_1^{ijk} = \sum_{l \in L} x_2^{jkl} \qquad (j,k) \in J \times K \qquad \text{(pool mass balance)}$$

$$\sum_{i \in I} q_1^{ijkp} x_1^{ijk} = q_2^{jkp} \sum_{l \in L} x_2^{jkl} \qquad (j,k,p) \in J \times K \times P \qquad \text{(pool quality balance) nonlinear}$$

$$\sum_{j \in J} \sum_{k \in K} x_2^{jkl} = x_S^l \qquad l \in L \qquad \text{(product mass balance)}$$

$$x_S^l \leq D^l \qquad l \in L \qquad \text{(maximum product demand)}$$

$$\sum_{j \in J} \sum_{k \in K} q_2^{jkp} x_2^{jkl} \leq Z^{lp} x_S^l \qquad (l,p) \in L \times P \qquad \text{(product quality requirement) nonlinear}$$

$$x_F^{ij}, x_1^{ijk}, x_2^{jkl}, x_S^l \geq 0, q_2^{jkp} \in \mathbb{R}$$

Figure 6.2 displays the pooling structure of a case study with corresponding data for crudes (A, B), blends (X, Y), distillation cuts (M, N), and final products (P, Q, R, S). Two different qualities are considered in this case.

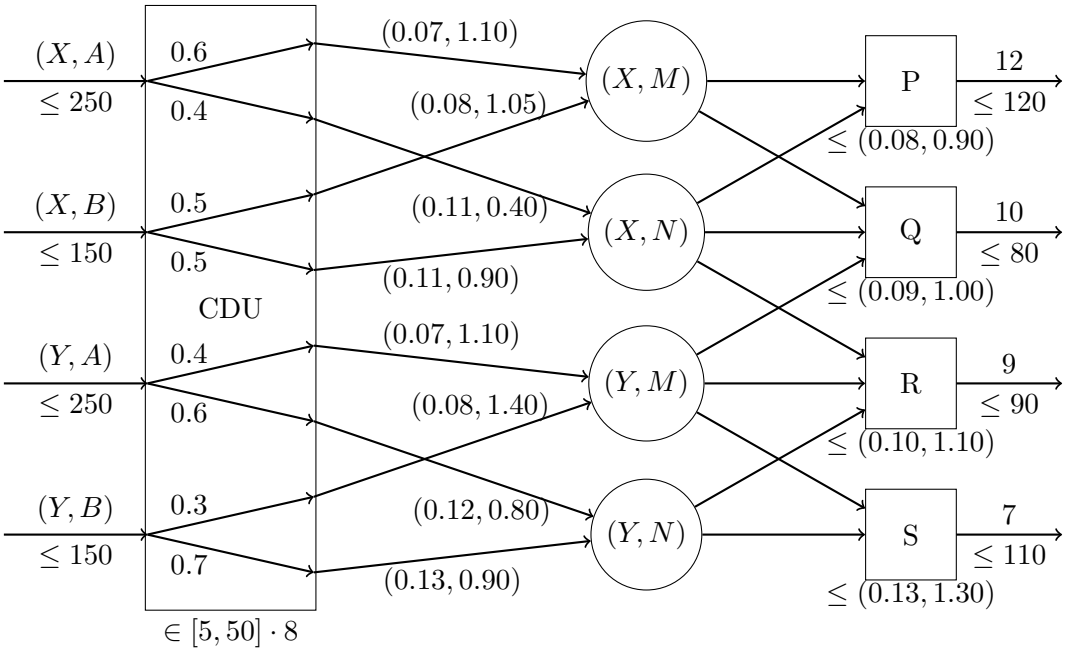


Figure 6.2: Refinery planning case study.

In the remainder of the chapter, we consider the following NLP, which is a simplified version of the planning model (P_P).

$$(P_P) \quad \begin{cases} \max & V_P^T x_S \\ \text{s.t.} & f_P(x_F, x_I, x_S) \leq 0 \\ & g_P(x_I) \leq 0 \\ & x_F \in \mathbb{R}^{|F|}, x_I \in \mathbb{R}^{|I|}, x_S \in \mathbb{R}^{|S|} \end{cases}$$

The nomenclature used is as follows:

- V_P is the market value of final products
- x_F is a set of continuous variables representing CDU feedstock quantities over the single planning period
- x_S is a set of continuous variables representing final products sales
- x_I is a set of intermediate continuous variables (e.g. pool quantity and property variables)

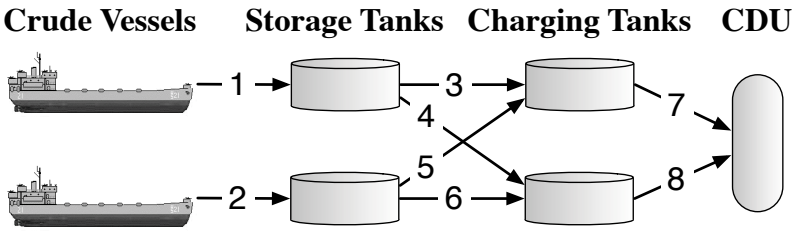


Figure 6.3: Refinery crude-oil scheduling system for COSP1.

- $f_P(x_F, x_I, x_S) \leq 0$ is the set of *linear* constraints (e.g. material balance constraints)
- $g_P(x_I) \leq 0$ is the set of *nonlinear* constraints (e.g. quality balance constraints)

6.2.2 Crude-Oil Scheduling Problem

The crude-oil scheduling problem deals with the unloading, transfer and blending operations executed on crude-oil tankers and crude-oil inventories. The goal is to sequentially prepare multiple crude blends, which are defined by specific property requirements. Each type of crude blend corresponds to a specific CDU operating mode. Different objectives have been studied, namely minimization of logistics costs (see chapter 5 and Lee et al., 1996) or maximization of profit (see chapters 3 and 4). In this work, the objective is to minimize the total replacement cost of the crudes that are selected for distillation. The replacement cost is the cost of replacing the crude once it has been processed. The crude-oil schedule must satisfy inventory capacity limitations, crude tankers arrival dates as well as the logistics constraints described in chapter 3. Figure 6.3 shows the refinery system corresponding to problem 1 introduced in Lee et al. (1996). Table 6.1 displays the dimensionless data for this example. Besides a different objective function, the example is modified by introducing a minimum duration of one day for distillation operations. Therefore, due to crude blend alternative sequencing, at most 4 batches of each crude mix can be processed in 8 days. The scheduling problem is formulated using the MOS time representation as introduced in chapter 2 (see appendix C).

In the remainder of the chapter, we consider the following MINLP, which is a simplified

Table 6.1: Crude-oil scheduling data for case study.

Scheduling horizon			8 days
Vessels	Arrival time	Composition	Amount of crude
Vessel 1	0	100% A	100
Vessel 2	4	100% B	100
Storage tanks	Capacity	Initial composition	Initial amount of crude
Tank 1	[0, 100]	100% A	25
Tank 2	[0, 100]	100% B	75
Charging tanks	Capacity	Initial composition	Initial amount of crude
Tank 1 (mix X)	[0, 100]	80% A, 20% B	50
Tank 2 (mix Y)	[0, 100]	20% A, 80% B	50
Crudes	Property 1 (sulfur concentration)		Crude unit cost
Crude A	0.01		7
Crude B	0.06		6
Crude mixtures	Property 1 (sulfur concentration)		Maximum number of batches
Crude blend X	[0.015, 0.025]		4
Crude blend Y	[0.045, 0.055]		4
Unloading flowrate	[0, 50]	Transfer flowrate	[0, 50]
Distillation flowrate	[5, 50]	Minimum duration of distillations	1 day

version of the scheduling model (P_S).

$$(P_S) \quad \begin{cases} \max & -V_C^T y_F \\ \text{s.t.} & f_S(y_B, y_C, y_F) \leq 0 \\ & g_S(y_C) \leq 0 \\ & y_B \in \{0,1\}^{|B|}, y_C \in \mathbb{R}^{|C|}, y_F \in \mathbb{R}^{|F|} \end{cases}$$

The nomenclature used is as follows:

- V_C is the replacement cost of crude-oils (usually based on market value)
- y_F is a set of continuous variables representing total CDU feedstock quantities over the scheduling horizon
- y_C is a set of continuous variables representing other continuous decisions (e.g. timing decisions)
- y_B is a set of binary variables representing the sequencing decisions (i.e. variables Z_{iv})
- $f_S(y_B, y_C, y_F) \leq 0$ is the set of *linear* constraints (e.g. scheduling constraints)
- $g_S(y_C) \leq 0$ is the set of *nonlinear* stream composition constraints (i.e. constraint (3.6c))

6.2.3 Full-Space Problem

Given the importance of crude selection for refinery optimization, the refinery planning problem and the crude-oil scheduling problem should ideally be optimized simultaneously. This can only be done by solving an integrated *full-space* MINLP problem, denoted (P), which aims at optimizing all refinery decisions subject to planning, scheduling, and linking constraints.

$$(P) \quad \begin{cases} \max & V_P^T x_S - V_C^T y_F \\ \text{s.t.} & f_P(x_F, x_I, x_S) \leq 0 \\ & g_P(x_I) \leq 0 \\ & f_S(y_B, y_C, y_F) \leq 0 \\ & g_S(y_C) \leq 0 \\ & y_F - x_F = 0 \\ & x_F \in \mathbb{R}^{|F|}, x_I \in \mathbb{R}^{|I|}, x_S \in \mathbb{R}^{|S|} \\ & y_B \in \{0,1\}^{|B|}, y_C \in \mathbb{R}^{|C|}, y_F \in \mathbb{R}^{|F|} \end{cases}$$

The integrated objective is to maximize profit defined by final product sales revenues minus crude-oil replacement costs. The linking constraint $y_F - x_F = 0$ ensures consistency of planning and scheduling decisions in terms of CDU feedstock quantities. More precisely, it ensures that the amounts of crudes selected for distillation are identical in the planning and scheduling solutions. Also, to be consistent in time, it is considered that the planning and scheduling horizons have identical lengths.

In the remainder of the chapter, we use the notation $v\,(P)$ to denote the optimal objective value for problem (P).

6.3 Lagrangian Decomposition Scheme

The full-space problem (P) is a large-scale MINLP, which contains many binary variables from the crude-oil scheduling problem and many non-convex constraints from the refinery planning model. Due to convergence issues and the presence of many potential local optima,

standard MINLP solvers for convex optimization, such as AlphaECP, Bonmin, DICOPT, KNITRO, or SBB, may fail solving the model or may return poor solutions. Global MINLP solvers, such as BARON, Couenne, or LINDOGLOBAL, are in principle able to solve the problem but they may require prohibitive computational times. Therefore, a specific solution strategy needs to be developed to address this problems.

Robertson et al. (2010) proposed a multi-level approach consisting of approximating the refinery planning model by multiple linear regressions that are then used in the crude-oil scheduling problem for the minimization of the total logistics and production costs. The method is applied to a case study comprising two different crudes. Although computationally effective, the use of linear regressions may not be sufficient for the the global optimization of highly nonlinear refinery planning model.

In this work, we present an integration approach based on *Lagrangian decomposition*, which is a special case of *Lagrangian relaxation* (Guignard, 2003). The idea is to build a relaxed version of the full-space problem, which is decomposable, and therefore much easier to solve. In particular, the decomposition procedure is based on the dualization of the linking constraint $y_F - x_F = 0$. The relaxed problem $\big(P_R(\lambda)\big)$, composed of NLP and MINLP models, is defined by removing this constraint and penalizing its violations by adding the term $\lambda^T(y_F - x_F)$ to the objective function. The parameter λ is a *Lagrange multiplier* whose value is fixed prior to solving the model and adjusted iteratively.

$$\big(P_R(\lambda)\big) \quad \begin{cases} \max & V_P^T x_S - \lambda^T x_F + (\lambda - V_C)^T y_F \\ \text{s.t.} & f_P(x_F, x_I, x_S) \leq 0 \\ & g_P(x_I) \leq 0 \\ & f_S(y_B, y_C, y_F) \leq 0 \\ & g_S(y_C) \leq 0 \\ & x_F \in \mathbb{R}^{|F|}, x_I \in \mathbb{R}^{|I|}, x_S \in \mathbb{R}^{|S|} \\ & y_B \in \{0,1\}^{|B|}, y_C \in \mathbb{R}^{|C|}, y_F \in \mathbb{R}^{|F|} \end{cases}$$

As already mentioned, problem $\big(P_R(\lambda)\big)$ is easier to solve as it can be decomposed into two

subproblems $\big(P_P(\lambda)\big)$ and $\big(P_S(\lambda)\big)$.

$$v\left(P_R(\lambda)\right) = v\left(P_P(\lambda)\right) + v\left(P_S(\lambda)\right)$$

The subproblem $\big(P_P(\lambda)\big)$, an NLP, is a modification of the original refinery planning problem $\big(P_P\big)$ as it consists of assigning crude costs λ to the CDU feedstock variables x_F. For a given crude i, increasing λ^i will decrease the incentive to select this crude for distillation processing. On the other hand, decreasing λ^i will increase the incentive to select it.

$$\big(P_P(\lambda)\big) \quad \begin{cases} \max & V_P^T x_S - \lambda^T x_F \\ \text{s.t.} & f_P(x_F, x_I, x_S) \leq 0 \\ & g_P(x_I) \leq 0 \\ & x_F \in \mathbb{R}^{|F|}, x_I \in \mathbb{R}^{|I|}, x_S \in \mathbb{R}^{|S|} \end{cases}$$

The subproblem $\big(P_S(\lambda)\big)$, an MINLP, is a modification of the original crude-oil scheduling problem (P_S) as it consists of assigning crude values λ to the CDU feedstock variables y_F. For a given crude i, increasing λ^i will increase the incentive to select this crude for blending and distillation processing. On the other hand, decreasing λ^i will decrease the incentive to select it.

$$\big(P_S(\lambda)\big) \quad \begin{cases} \max & (\lambda - V_C)^T y_F \\ \text{s.t.} & f_S(y_B, y_C, y_F) \leq 0 \\ & g_S(y_C) \leq 0 \\ & y_B \in \{0,1\}^{|B|}, y_C \in \mathbb{R}^{|C|}, y_F \in \mathbb{R}^{|F|} \end{cases}$$

On the whole, the Lagrange multiplier λ can be seen as a crude purchase cost for the planning system, and as a crude sales value for the scheduling system. Therefore, the spatial Lagrange decomposition procedure applied to this problem can be seen as introducing a crude market between the planning and scheduling systems (see Fig. 6.4). The planning system acts as a consumer who buys crude from the market, while the scheduling system acts as a producer who sells it to the market. It is clear that, for fixed prices, both actors

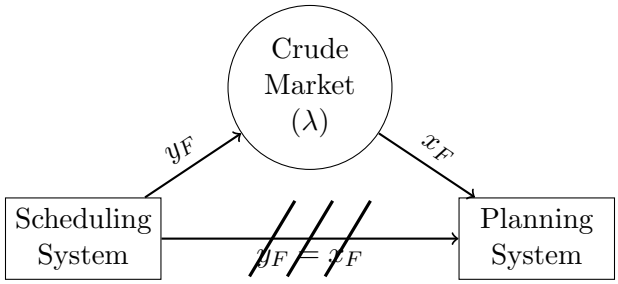


Figure 6.4: Economic interpretation of the Lagrangian decomposition.

are independent, which explains why the two corresponding subproblems can be solved in parallel.

Although computationally convenient, this decomposition procedure does not solve the original full-space problem. In particular, it is well-known that $\big(P_R(\lambda)\big)$ is a relaxation of the full-space problem, thus $v\left(P_R(\lambda)\right) > v\left(P\right)$. However, one can search for a Lagrange multiplier λ that minimizes $v\left(P_R(\lambda)\right)$ in order to get as close as possible to $v\left(P\right)$. This problem is called the *dual problem* and the function $\lambda \mapsto v\left(P_R(\lambda)\right)$ is often called the *Lagrangian function*.

$$\big(P_D\big) \quad \min_{\lambda} v\left(P_R(\lambda)\right)$$

Duality theory establishes that $v\left(P_D\right) - v\left(P\right) \geq 0$ (see Guignard, 2003). In some cases (e.g. nonconvex models), we may have $v\left(P_D\right) - v\left(P\right) > 0$ and this difference is called *dual gap*. Our hope is that this dual gap is small enough so that valuable information can be inferred to generate near-optimal heuristic solutions (see section 6.5).

6.4 Solution of the Dual Problem

Several approaches have been proposed in the literature in order to solve the dual problem associated with the Lagrangian relaxation. A classical approach is the *subgradient method* proposed by Held and Karp (1971) and Held and Karp (1974). Many researchers have used and improved this technique over the years (see Camerini et al., 1975; Bazaraa and Sherali,

1981; Fisher, 1981). This approach is preferred as it usually predicts very good Lagrange multiplier updates. However, special care must be taken in order to ensure convergence and it requires a good strategy for defining and updating the subgradient step size. Another approach that theoretically displays better convergence properties has been introduced by Cheney and Goldstein (1959) and Kelley (1960). It is often denoted as the *cutting plane method*. In practice, this approach usually takes a long time to converge as many iterations are necessary in order to obtain good Lagrange multiplier updates. A refinement of this approach is the *boxstep method* presented in Marsten et al. (1975). It allows obtaining better updates for the Lagrange multiplier during early iterations while keeping the same convergence properties. Other refinements of previous approaches include the *bundle method* (Lemaréchal, 1974), the *volume algorithm* (Barahona and Anbil, 2000), and the *analytic center cutting plane method* (Goffin et al., 1992).

All the above approaches are based on an iterative solution procedure between the primal and the dual world. Figure 6.5 gives a schematic description of this algorithm. The first step consists of initializing the Lagrange multipliers. Problem-specific strategies, often based on the economic interpretation of λ, exist in order to provide good initial values. Then, at each iteration the relaxed problem is solved and a primal solution is obtained. If a stopping criterion is satisfied, the algorithm converges. Otherwise, the Lagrange multipliers are updated for the next iteration. The definition of the stopping criterion depends on the approach used and can, in certain cases, ensure convergence to an optimal dual solution λ^*.

In this work, we introduce a new hybrid method to update the Lagrange multipliers. It is based on the three concepts of cutting planes, subgradient and boxstep. Cutting planes are valid constraints for the dual problem that are generated at each iteration. They are used to record valuable dual information to be used during later iterations. A subgradient provides a descent direction for the dual problem, while a boxstep is defined to allow deviations from this direction within a specified domain. The combination of these techniques ensures good convergence properties while providing efficient Lagrange multiplier updates. At iteration $K + 1$, the Lagrange multiplier is updated to the solution of the following restricted LP

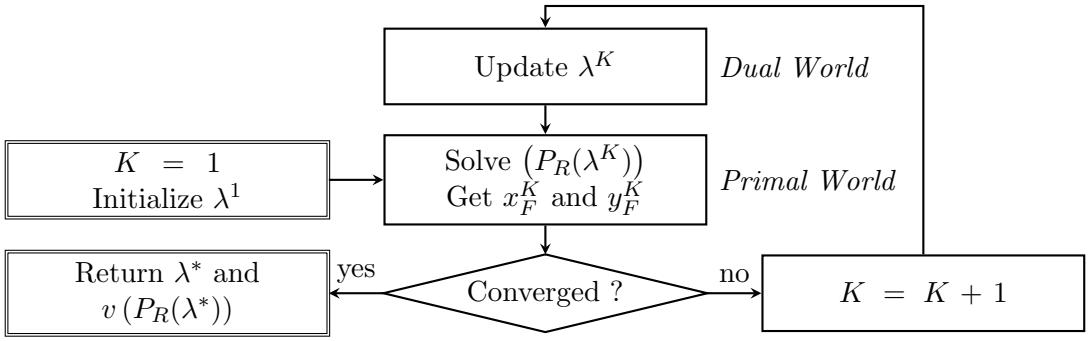


Figure 6.5: General iterative primal-dual algorithm.

dual problem with subgradient-based boxstep.

$$\left(\hat{P}_D^{K+1}\right) \quad \begin{cases} \min & \eta \\ \text{s.t.} & \eta \geq V_P^T x_S^k - V_C^T y_F^k + \lambda^T (y_F^k - x_F^k) \qquad \forall k = 1 \dots K \qquad (\text{CP}^k) \\ & \lambda = \lambda^K + \alpha \frac{v(P_D) - v\left(P_R(\lambda^K)\right)}{\left|\left|y_F^K - x_F^K\right|\right|^2} (y_F^K - x_F^K) + \delta \qquad (\text{SG+BS}) \\ & \eta \in \mathbb{R}, \lambda \in \mathbb{R}^{|F|} \\ & \alpha \in]-\infty, \overline{\alpha}], \delta \in [-\overline{\delta}, \overline{\delta}]^{|F|} \end{cases}$$

The variables λ and η are classically used in the pure cutting plane approach. The pure cutting plane restricted LP dual problem $\left(P_D^{K+1}\right)$ consists of minimizing η subject to constraints (CP^k), $k = 1 \dots K$ only:

$$\left(P_D^{K+1}\right) \quad \begin{cases} \min & \eta \\ \text{s.t.} & \eta \geq V_P^T x_S^k - V_C^T y_F^k + \lambda^T (y_F^k - x_F^k) \quad \forall k = 1 \dots K \quad (\text{CP}^k) \\ & \eta \in \mathbb{R}, \lambda \in \mathbb{R}^{|F|} \end{cases}$$

This problem is always unbounded during early iterations (i.e. $v\left(P_D^K\right) = -\infty$) so it cannot be used directly to update the Lagrange multipliers. This issue can be solved by defining a bounded feasible set for the multipliers based on their interpretation (e.g. lower and upper bounds). However, in this work, an iteratively updated boxstep based on the subgradient step is used so that the restricted dual problem $\left(\hat{P}_D^{K+1}\right)$ is bounded.

The subgradient step is classically defined as

$$\lambda = \lambda^K + \alpha \left(v\left(P_D\right) - v\left(P_R(\lambda^K)\right)\right) / \left|\left|y_F^K - x_F^K\right|\right|^2 (y_F^K - x_F^K)$$

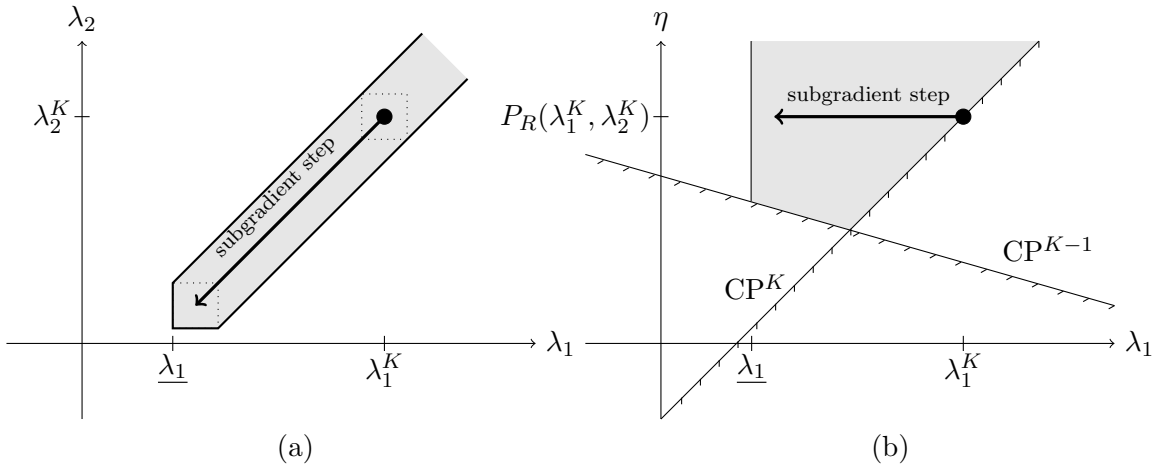


Figure 6.6: Plots of the feasible space of $(\hat{P}_D^{K+1})$.

where $v(P_D)$ can be estimated using a heuristic solution for (P). However, instead of heuristically updating the step size, it is optimized using variable α, which is bounded by the parameter $\overline{\alpha} > 0$. Note that α is allowed to take negative values. Variable δ defines a deviation from the subgradient step. The parameter $\overline{\delta} > 0$ is the maximum deviation in each multiplier direction and defines a full-dimensional boxstep. Both subgradient and boxstep concepts are simultaneously embedded in constraint (SG+BS). Note that the parameters $\overline{\alpha}$ and $\overline{\delta}$ can be heuristically updated at each iteration. In practice, our computational experiments have shown that using fixed values for $\overline{\alpha}$ and $\overline{\delta}$ is a reasonable choice.

Figure 6.6 displays the feasible space (grey area) of $(\hat{P}_D^{K+1})$. The projection on the space of Lagrange multipliers (λ_1, λ_2) depicts the shape of the subgradient-based boxstep. In the space of (λ_1, η), CP^k represents the projection of the cutting plane generated at iteration k. Note that the feasible space of $\left(\hat{P}_D^{K+1}\right)$ contains $\left(\lambda_1 = \lambda_1^K, \lambda_2 = \lambda_2^K, \eta = v\left(P_R(\lambda_1^K, \lambda_2^K)\right)\right)$. In both plots (a) and (b), $\underline{\lambda_1}$ corresponds to a lower bound on multiplier λ_1 induced by the boxstep constraints.

The stopping criterion for this hybrid strategy is identical to the pure cutting plane method and is based on the *Lagrangian gap* between the relaxed primal problem and the

restricted dual problem:

$$v\left(P_R(\lambda^K)\right) - v\left(P_D^K\right) \leq \varepsilon \tag{6.1}$$

In the pure cutting plane approach (without constraint (SG+BS)), the optimal value of the restricted dual problem $v\left(P_D^K\right)$ iteratively approximates $v\left(P_D\right)$ from below since it involves the minimization of a relaxation of $v\left(P_D\right)$.

$$v\left(P_D^K\right) \leq v\left(P_D\right) \quad \forall K \tag{6.2}$$

Therefore, the stopping criterion (6.1) ensures convergence to an ε-optimal solution of the dual problem $\left(P_D\right)$ as:

$$v\left(P_R(\lambda^K)\right) - v\left(P_D^K\right) \leq \varepsilon \Rightarrow v\left(P_R(\lambda^K)\right) - v\left(P_D\right) \leq \varepsilon \tag{6.3}$$

In the proposed approach, when the pure restricted dual problem $\left(P_D^K\right)$ becomes bounded, the stopping criterion (6.1) is used to check convergence while the hybrid restricted dual problem $\left(\hat{P}_D^K\right)$ is used to update the Lagrange multipliers. Finite convergence properties for the pure cutting plane and boxstep methods in the context of mixed-integer linear programming can be obtained in Frangioni (2005) and Marsten et al. (1975), respectively. For the rest of the chapter, it is assumed that *practical* convergence of the proposed hybrid method can be achieved in the context of integrated refinery planning and scheduling.

6.5 Heuristic Solutions

In this section, a classical adaptation of the primal-dual algorithm is presented in order to obtain solutions that satisfy all constraints of the full-space problem, including linking constraints. As explained by Frangioni (2005), the solution of the Lagrangian dual problem yields primal information that can be used to generate good heuristic solutions for $\left(P\right)$. In this chapter, a heuristic step is introduced in the iterative algorithm to produce valid lower bounds P^{LB} (see Fig. 6.7). This induces a second stopping criterion based on the dual gap:

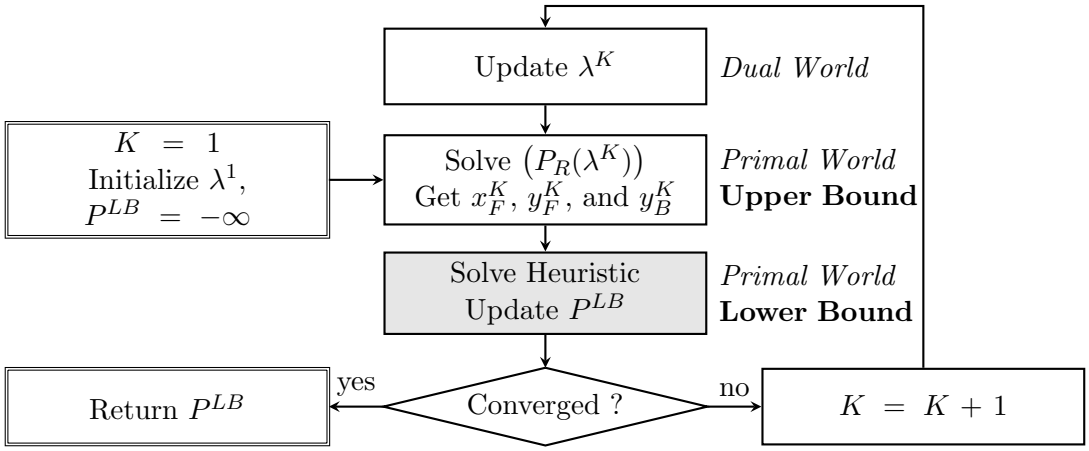


Figure 6.7: Iterative primal-dual algorithm with heuristic step.

$v\left(P_R(\lambda^K)\right) - P^{LB} \leq \varepsilon$. If either of the two stopping criteria is satisfied, the algorithm converges and returns P^{LB}.

The heuristic algorithm consists of fixing binary variables y_B from the crude-oil scheduling formulation to their values y_B^K in the solution of the relaxed problem $\left(P_R(\lambda^K)\right)$. As a consequence, the full-space problem $\left(P\right)$ reduces to a continuous NLP, denoted $\left(P_H(y_B^K)\right)$, and can then be effectively solved. If it is feasible and its (local) optimal solution is better than the previous incumbent, P^{LB} is updated. Otherwise, P^{LB} is left unchanged.

$$\left(P_H(y_B^K)\right) \quad \begin{cases} \max & V_P^T x_S - V_C^T y_F \\ \text{s.t.} & f_P(x_F, x_I, x_S) \leq 0 \\ & g_P(x_I) \leq 0 \\ & f_S(y_B^K, y_C, y_F) \leq 0 \\ & g_S(y_C) \leq 0 \\ & y_F - x_F = 0 \\ & x_F \in \mathbb{R}^{|F|}, x_I \in \mathbb{R}^{|I|}, x_S \in \mathbb{R}^{|S|} \\ & y_C \in \mathbb{R}^{|C|}, y_F \in \mathbb{R}^{|F|} \end{cases}$$

In this heuristic solution, fixing binary variables y_B to y_B^K corresponds to fixing the selection and sequencing of operations for the crude-oil scheduling system. Therefore, when

solving problem $\left(P_H(y_B^K)\right)$, the nonlinear solver has the opportunity to re-optimize all other continuous decisions such as quantities, blend recipes, and timing of operations (start time, duration, and end time). It is crucial to note that the timing of operations can only be re-optimized if these decisions are handled by continuous variables. For instance, the discrete-time MOS-FST representation (see chapter 2), uses binary variables to determine the timing decisions whereas all other representations, denoted by MOS, MOS-SST, and SOS, use continuous variables instead. This shows a clear advantage, although not intuitive, of continuous-time scheduling formulations over discrete-time representations. In particular, the latter might be inefficient in the context of this work as it would decrease the flexibility of the heuristic algorithm to find good, or at least feasible solutions.

Overall, it is interesting to note that using this approach, the iterative primal-dual algorithm that solves the Lagrangian dual problem acts as a *discrete solution generator* that suggests potentially good discrete solutions for the full-space problem. In other words, it searches the optimal selection and sequencing of operations for the crude-oil scheduling system. Additionally, it provides an upper bound for the global optimal solution.

6.6 Remarks

6.6.1 CDU Feedstocks and Lagrange Multipliers

The number of Lagrange multipliers highly depends on the CDU feedstock possibilities. Exactly one multiplier is needed for each feasible combination of crude and type of crude blend (or corresponding CDU operating mode). In the case study, there are 2 different crudes which can both be blended in any of the 2 different types of blends, so 4 Lagrange multipliers are needed to solve the dual problem. Typical large-scale refineries may need 50 and up to 100 Lagrange multipliers.

The optimal value of the Lagrange multipliers correspond to the optimal marginal costs of the linking constraints for the convexified problem (for more details in the case of MILP models, see Frangioni, 2005). Therefore, the Lagrange multipliers can be seen as the optimal

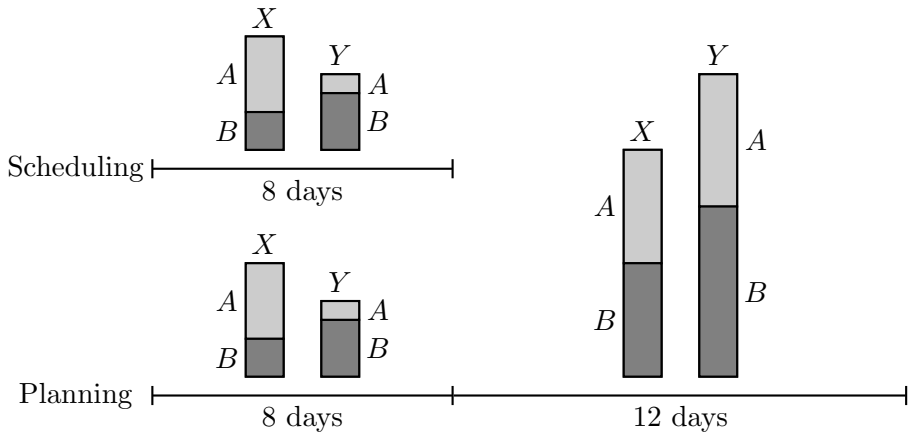


Figure 6.8: Crude-oil scheduling and multi-period refinery planning integration.

pricing strategy between the crude-oil scheduling system and the refinery planning system. In other words, the optimal Lagrange multipliers are crude prices for which it is marginally equivalent to either exchange crudes between the two systems or sell and buy to and from the crude market (see Fig. 6.4). From this observation, it is natural to use the crude costs defined in the crude-oil scheduling problem as initialization for the Lagrange multipliers. For the case study, we use the following initial values:

$$\lambda^1_{(X,A)} = \lambda^1_{(Y,A)} = 7$$
$$\lambda^1_{(X,B)} = \lambda^1_{(Y,B)} = 6$$

6.6.2 Multi-Period Refinery Planning

In this work, the refinery planning problem is expressed over a single period for which CDU feedstocks are synchronized with the crude-oil scheduling problem. Even though it is often computationally critical to increase the time horizon for scheduling problem, this can easily be done in refinery planning models by introducing additional time periods. Therefore, one can define 2 or more time periods for the refinery planning model and synchronize CDU feedstocks for the first period only, as shown in Figure 6.8. In this particular case, the

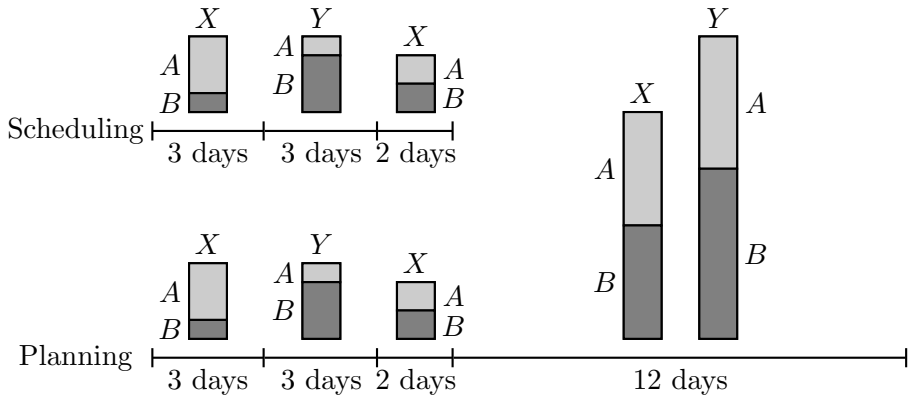


Figure 6.9: Disaggregated CDU feedstocks synchronization.

refinery planning decisions for the second period, including CDU feedstock decisions x_F, are made without taking into account crude-oil scheduling constraints. This methodology allows making short-term scheduling decision while considering the long-term economic impacts of these decisions, which cannot be done with detailed long-term scheduling models due to their computational complexity.

6.6.3 CDU Feedstocks Aggregation

An important issue with the proposed approach comes from the fact that the CDU feedstocks for the linking period are aggregated. In the optimal solution, the crude-oil operations schedule prepares several *batches* for each type of crude blends. Then, for each of these blend types, all the corresponding batches are accumulated and the refinery planning solution determines the processing decisions for the *aggregated batch*. This approximation may lead to sub-optimality or even technical infeasibility of the solutions obtained. This problem can be solved by postulating exactly one period for each batch and synchronizing all the corresponding CDU feedstocks. Figure 6.9 depicts the synchronization of disaggregated CDU batches.

6.6.4 Handling Nonlinearities in Crude-Oil Scheduling Model

Although, the relaxed problem $\left(P_R(\lambda)\right)$ is decomposable, it is not easy to solve to global optimality. In particular, two major issues arise. First, the crude-oil scheduling problem $\left(P_S(\lambda)\right)$ corresponds to an MINLP due to the presence of nonlinear composition constraints. In chapter 3, an MILP relaxation is derived by simply dropping these nonlinear constraints. The solution can then be refined by fixing the binary variables and solving the reduced NLP, similarly to the heuristic approach presented in section 6.5. Results show that the solution obtained is close to the global optimum as it tends to satisfy the relaxed nonlinear constraints. Therefore a similar methodology is used in this work. Instead of simply dualizing the linking constraints, the nonlinear scheduling constraints $g_S(y_C) \leq 0$ are also relaxed (i.e. dropped). The modified relaxed full-space MINLP problem (NLP + MILP) is denoted by $\left(\tilde{P}_R(\lambda)\right)$:

$$\left(\tilde{P}_R(\lambda)\right) \quad \begin{cases} \max & V_P^T x_S - \lambda^T x_F + (\lambda - V_C)^T y_F \\ \text{s.t.} & f_P(x_F, x_I, x_S) \leq 0 \\ & g_P(x_I) \leq 0 \\ & f_S(y_B, y_C, y_F) \leq 0 \\ & x_F \in \mathbb{R}^{|F|}, x_I \in \mathbb{R}^{|I|}, x_S \in \mathbb{R}^{|S|} \\ & y_B \in \{0,1\}^{|B|}, y_C \in \mathbb{R}^{|C|}, y_F \in \mathbb{R}^{|F|} \end{cases}$$

The corresponding modified crude-oil scheduling MILP subproblem is denoted by $\left(\tilde{P}_S(\lambda)\right)$:

$$\left(\tilde{P}_S(\lambda)\right) \quad \begin{cases} \max & (\lambda - V_C)^T y_F \\ \text{s.t.} & f_S(y_B, y_C, y_F) \leq 0 \\ & y_B \in \{0,1\}^{|B|}, y_C \in \mathbb{R}^{|C|}, y_F \in \mathbb{R}^{|F|} \end{cases}$$

The decomposability property is preserved:

$$v\left(\tilde{P}_R(\lambda)\right) = v\left(P_P(\lambda)\right) + v\left(\tilde{P}_S(\lambda)\right)$$

Finally, the modified dual problem $\left(\tilde{P}_D\right)$ can be defined as:

$$\left(\tilde{P}_D\right) \quad \min_{\lambda} v\left(\tilde{P}_R(\lambda)\right)$$

This modified dual problem still provides a valid upper bound for the original full-space problem $\big(P\big)$. The following modified heuristic problem $\big(\tilde{P}_H(y_B^K)\big)$ is also defined. It is obtained from the original heuristic problem $\big(P_H(y_B^K)\big)$ by dropping the nonlinear scheduling constraints. It is used as a first heuristic step to get a good initial point before solving the original heuristic NLP problem $\big(P_H(y_B^K)\big)$.

$$\big(\tilde{P}_H(y_B^K)\big) \quad \begin{cases} \max & V_P^T x_S - V_C^T y_F \\ \text{s.t.} & f_P(x_F, x_I, x_S) \leq 0 \\ & g_P(x_I) \leq 0 \\ & f_S(y_B^K, y_C, y_F) \leq 0 \\ & y_F - x_F = 0 \\ & x_F \in \mathbb{R}^{|F|}, x_I \in \mathbb{R}^{|I|}, x_S \in \mathbb{R}^{|S|} \\ & y_C \in \mathbb{R}^{|C|}, y_F \in \mathbb{R}^{|F|} \end{cases}$$

6.6.5 Handling Nonlinearities in the Refinery Planning Model

In order to obtain valid upper bounds when solving $\big(P_R(\lambda)\big)$ or $\big(\tilde{P}_R(\lambda)\big)$, the refinery planning problem $\big(P_P(\lambda)\big)$ has to be solved to global optimality. Although global optimization of industrial large-scale refinery planning models is still unachievable, the refinery planning case study presented in section 6.2 is solvable by the global NLP solver BARON in reasonable time. However, it should be noted that it is critical to provide tight bounds for the quality variables q_2^{jkp}. In particular, based on the structure of the pooling system, we use the following bounds:

$$\min_{i \in I} q_1^{ijkp} \leq q_2^{jkp} \leq \max_{i \in I} q_1^{ijkp}$$

6.6.6 Detailed Implementation

Based on previous remarks to handle nonlinearities, the complete heuristic algorithm is developed as depicted in Figure 6.10. Although global optimality cannot be ensured, the dual gap can be estimated using the upper bound provided by $v\left(\tilde{P}_R(\lambda^*)\right)$. Local NLP

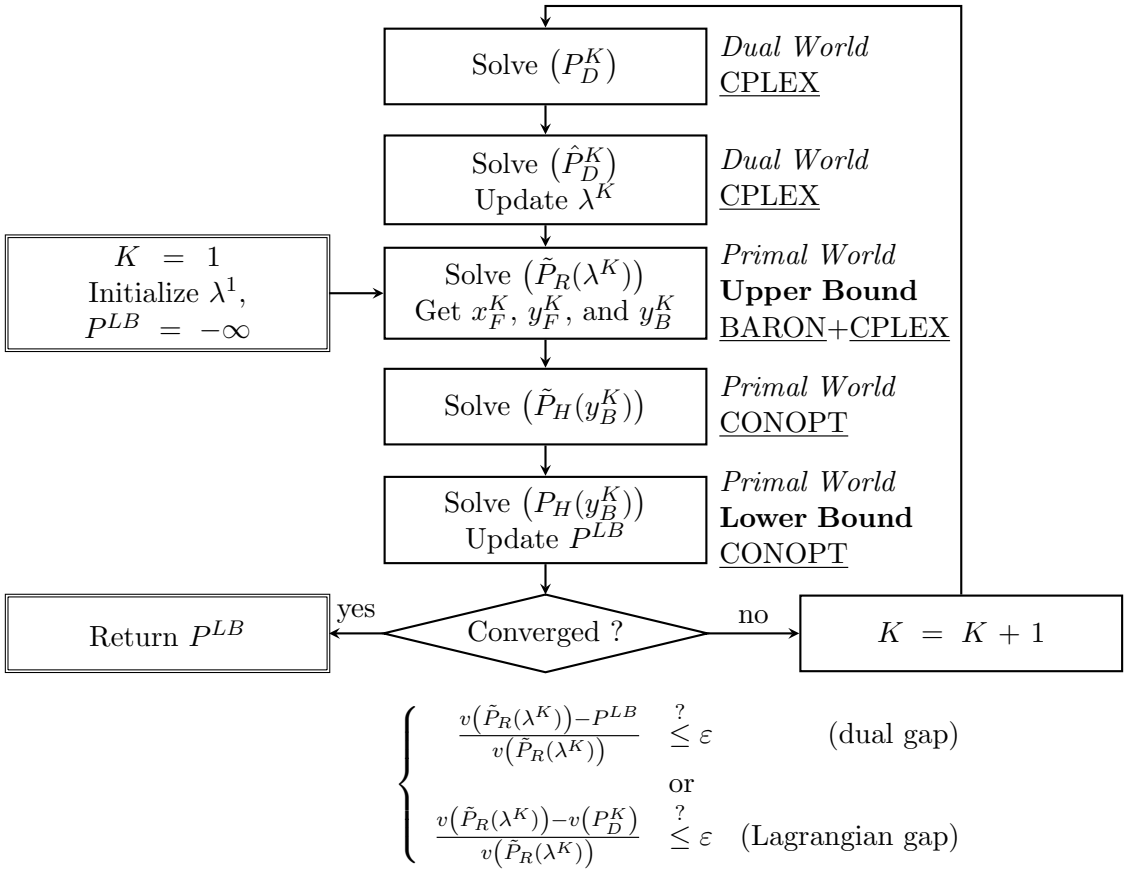


Figure 6.10: Complete algorithm implementation.

solvers, such as CONOPT, are used for the heuristic steps as the solution time is more critical than global optimality for these problems. The hybrid restricted dual problem $\left(\hat{P}_D^K\right)$ is solved using the best solution of the modified heuristic problems $\left(\tilde{P}_H(y_B^K)\right)$ to estimate the optimal dual solution $v\left(P_D\right)$. The stopping criterion is based on relative gaps but can also be expressed in terms of absolute gaps. Converging on the Lagrangian gap means that no further improvement of the upper bound can be achieved and the current Lagrange multipliers are optimal.

Table 6.2: Lagrangian iterations statistics (6 priority-slots, NLP=SNOPT).

Iteration	Pure Dual $v\left(P_D^K\right)$	Hybrid Dual $v\left(\hat{P}_D^K\right)$	Step Size α^K	Modified Relaxation $v\left(\tilde{P}_R(\lambda^K)\right)$	Modified Heuristic $v\left(\tilde{P}_H(y_B^K)\right)$	Original Heuristic $v\left(P_H(y_B^K)\right)$	CPU Time
1	—[a]	—[b]	—[b]	645.000	400.942	—[c]	3s
2	—[a]	389.942	1	689.049	564.000	560.500	9s
3	—[a]	547.929	1	637.063	592.368	592.368	13s
4	—[a]	582.790	1	603.377	592.368	592.368	18s
5	—[a]	591.172	1	609.526	586.191	—[c]	24s
6	—[a]	590.629	0.851	598.380	562.881	—[c]	29s
7	580.717	591.324	-0.617	600.171	586.191	—[c]	33s
8	592.368	592.785	0.390	594.774	586.191	—[c]	46s
9	592.369	592.369	1	592.372	592.368	592.368	50s

[a] unbounded LP

[b] not available

[c] locally infeasible NLP

6.7 Numerical Illustration

In this section, several computational results are presented for three approaches: the direct MINLP approach, a basic sequential scheduling-planning procedure and the proposed Lagrangian decomposition method. Experiments have been performed on an Intel Xeon 1.86GHz processor using GAMS as the modeling and algorithmic language. A 1,000 seconds time limit has been used for each run. The following local NLP solvers have been used: CONOPT, SNOPT and IPOPT. In our experiments, the convergence tolerance ε is set to 0.0001, the maximum step size parameter $\overline{\alpha}$ is set to 1 and the step bound parameter $\overline{\delta}$ to 0.05.

The number of priority-slots for the crude-oil scheduling model is set to 6 and 7. Tables 6.2 and 6.3 show iteration statistics for the Lagrangian decomposition method using SNOPT as the heuristic NLP solver. In particular, the optimal value of each problem solved is given as well as the optimal step size calculated by the hybrid restrict dual problem and the cumulative CPU time at the end of each iteration (e.g., for 6 priority-slots, the first iteration took 3 seconds). Dashes are used when the information is not available (Hybrid

Table 6.3: Lagrangian iterations statistics (7 priority-slots, NLP=SNOPT).

Iteration	Pure Dual $v\left(P_D^K\right)$	Hybrid Dual $v\left(\hat{P}_D^K\right)$	Step Size α^K	Modified Relaxation $v\left(\tilde{P}_R(\lambda^K)\right)$	Modified Heuristic $v\left(\tilde{P}_H(y_B^K)\right)$	Original Heuristic $v\left(P_H(y_B^K)\right)$	CPU Time
1	—	—	—	645.000	393.470	—	3s
2	—	381.562	1	751.494	568.077	568.077	10s
3	—	552.076	1	622.785	592.368	—	16s
4	—	574.735	1	614.178	592.368	<u>592.368</u>	22s
5	—	583.166	1	602.218	592.368	592.079	50s
6	—	588.279	1	617.268	592.368	592.079	56s
7	—	592.856	0.911	600.752	592.368	<u>592.368</u>	66s
8	—	592.835	0.517	595.264	592.368	—	81s
9	—	592.288	0.357	595.292	592.368	<u>592.368</u>	95s
10	592.368	592.369	1	592.369	592.368	592.079	101s

Dual and Step Size for the first iteration), when the problem is unbounded (Pure Dual during the first few iterations), or when it is locally infeasible (Original Heuristic in some iterations).

In each case, the global optimal solution is found (see underlined entries in the Original Heuristic column) and proved optimal. It can be noted that in some iterations the step size variable α is strictly lower than 1, which corresponds to cases where the pure subgradient multiplier update would violate some cutting planes generated at previous iterations. The proposed approach automatically overcomes this issue. The increase of CPU time between 6 and 7 priority-slots is mostly explained by the increase in size of the MILP scheduling model. Figures 6.11 and 6.12 plot the evolution of the objective value of various problems solved during the Lagrangian iterations.

The optimal value of the Lagrange multipliers is $\lambda^*_{(X,A)} = \lambda^*_{(X,B)} = \lambda^*_{(Y,A)} = \lambda^*_{(Y,B)} = 7$. Figures 6.13 and 6.14 plot the evolution of the Lagrange multipliers during the Lagrangian iterations. The proposed approach demonstrates its efficiency through stable updates and fast convergence of the Lagrange multipliers.

A basic sequential procedure is introduced to compare with the Lagrangian decomposition

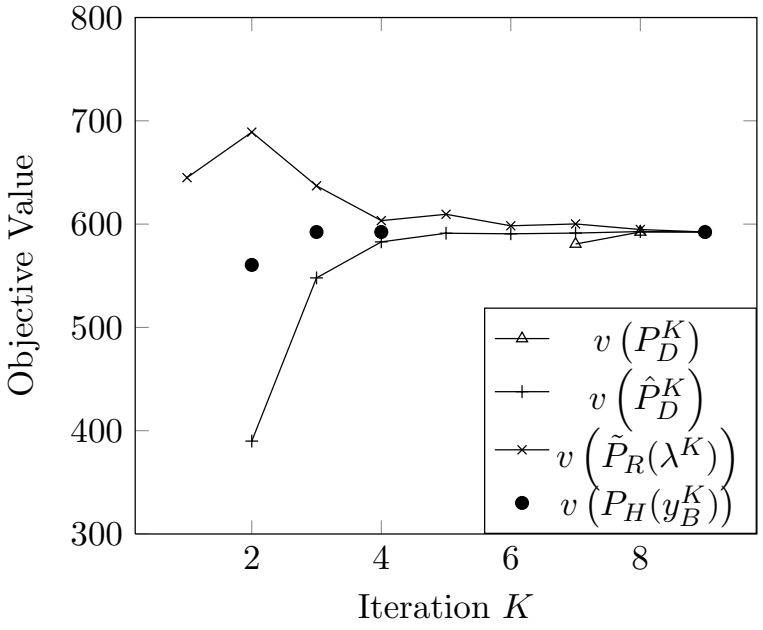


Figure 6.11: Lagrangian iteration objective values (6 priority-slots, NLP=SNOPT).

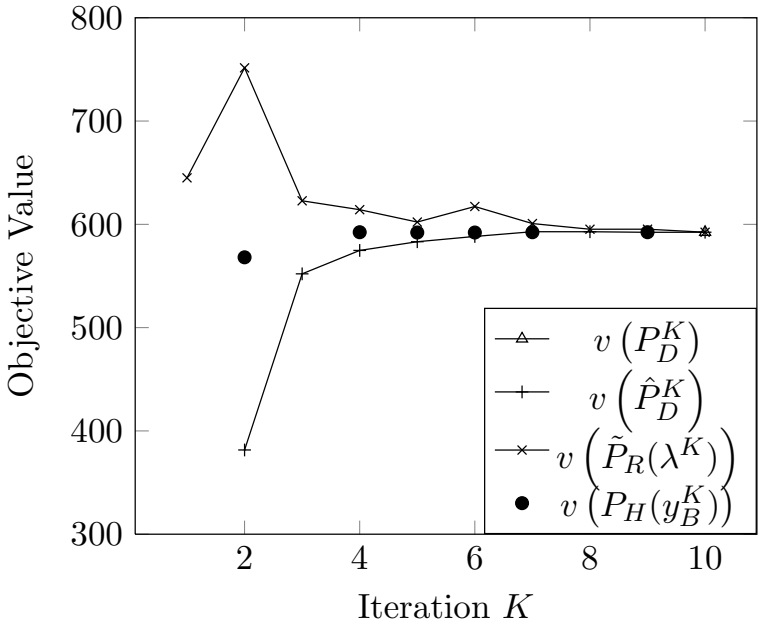


Figure 6.12: Lagrangian iteration objective values (7 priority-slots, NLP=SNOPT).

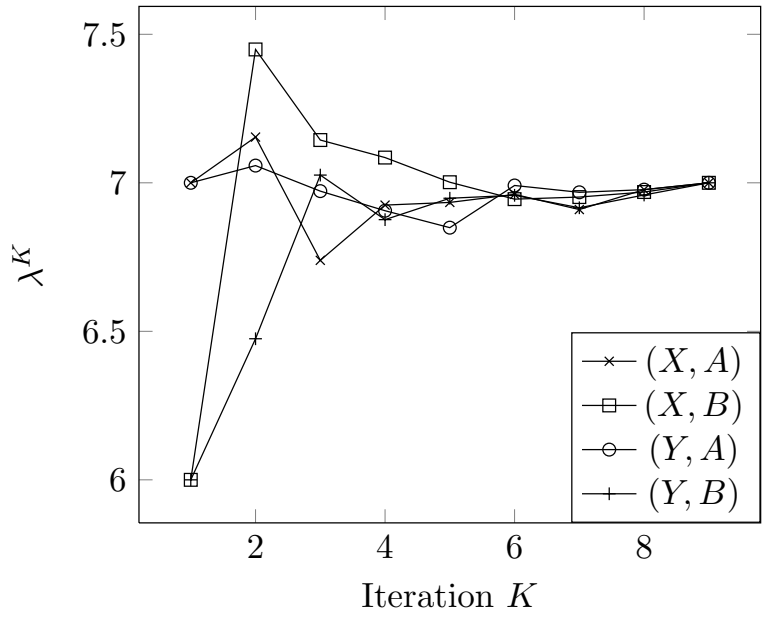


Figure 6.13: Lagrange multiplier updates (6 priority-slots, NLP=SNOPT).

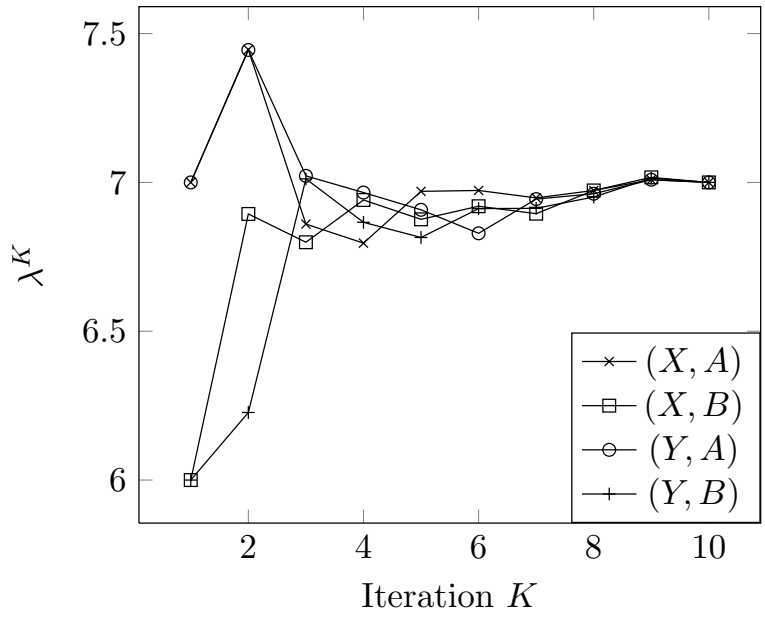


Figure 6.14: Lagrange multiplier updates (7 priority-slots, NLP=SNOPT).

Table 6.4: Comparative performance of different MINLP algorithms.

	6 priority-slots			7 priority-slots		
MINLP Solver	Objective Value	CPU Time	Optimality Gap	Objective Value	CPU Time	Optimality Gap
Proposed (CONOPT)	592.368	37s	0%	592.368	94s	0%
Proposed (SNOPT)	592.368	50s	0%	592.368	101s	0%
Proposed (IPOPT)	592.368	244s	0%	592.368	833s	0%
Sequential (BARON)	545.000	9s	—	545.000	10s	—
DICOPT (CONOPT)	545.000	5s	—	592.368	7s	—
DICOPT (SNOPT)	592.368	429s	—	592.368	6s	—
DICOPT (IPOPT)	568.077	54s	—	592.368	44s	—
AlphaECP (CONOPT)	512.324	67s	—	545.000	120s	—
AlphaECP (SNOPT)	512.324	67s	—	545.000	395s	—
AlphaECP (IPOPT)	512.324	69s	—	545.000	175s	—
SBB (CONOPT)	592.368	267s	—	592.368	+1,000s	—
SBB (SNOPT)	—	+1,000s	—	—	+1,000s	—
SBB (IPOPT)	—	+1,000s	—	493.536	+1,000s	—
LINDOGLOBAL	568.077	+1,000s	11.9%	532.857	+1,000s	17.1%
BARON	592.170	+1,000s	7.3%	400.000	+1,000s	37.5%

approach. First, the modified crude-oil scheduling problem $\left(\tilde{P}_S\right)$ is solved and the binary variables are fixed to their solution value. Then, the modified and original heuristic problems $\left(\tilde{P}_H(y_B^0)\right)$ and $\left(P_H(y_B^0)\right)$ are successively solved. This procedure is not computationally expensive as it requires solving only one MILP, solved with CPLEX, and two NLPs, both solved with BARON. However, the solution obtained, if any, might not be optimal.

Additionally, the Lagrangian decomposition approach is compared to the direct approach which consists of solving the full-space problem $\left(P\right)$ with various MINLP solvers. Table 6.4 shows the computational performance of these MINLP algorithms. The sequential approach quickly provides a feasible solution which is 8.0% lower than the global optimum (545.000 against 592,368). The solvers DICOPT, AlphaECP and SBB cannot guarantee global optimality of the solution returned while LINDOGLOBAL and BARON are actual global MINLP solvers. DICOPT is able to find the global optimal solution in many cases in reasonable CPU times. AlphaECP never finds the global optimum. SBB seems to return the best solutions when CONOPT is used as NLP solver but it is requires large CPU times.

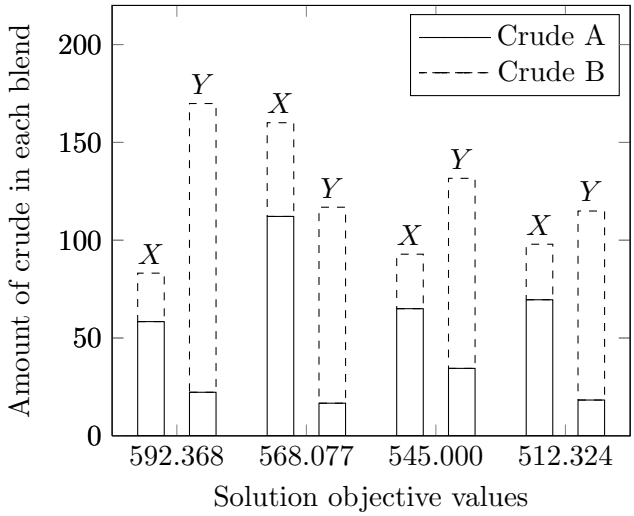


Figure 6.15: Blend compositions in solutions obtained with 6 priority-slots.

Neither LINDOGLOBAL or BARON have found the global optimum in the specified time limit. In comparison to these solvers, the proposed Lagrangian decomposition approach proves to be very effective for the following reasons:

- it is computationally effective (although DICOPT is faster);
- it always returns the global optimum;
- it is very robust with the choice of NLP solver (although IPOPT is significantly slower).

Figure 6.15 depicts the composition of each crude blend in four different solutions. Crude blend X is mostly composed of crude A while crude blend Y is mostly composed of crude B. If all solutions except the second one (with objective value 568.077) are considered, one may conclude that blend Y is "more profitable" than blend X because the objective value increases when processing larger amounts of blend Y and smaller amounts of blend X. Additionally, one could say that increasing the amount of distilled crude increases the overall profit (solution 1 processes 253.034 units of crude, solution 3, 224.49 units of crude, and solution 4, 212.867 units of crude). Interestingly, the second solution does not follow these observations. In this solution, the amount of blend X is larger the amount of blend Y.

Besides, the total amount of crude processed is the largest (276.923 units of crude). This shows how difficult it is to approximate the economic behavior of refinery operations, even for such a small case study. Therefore, it is possible that linear approximations of refinery operations as proposed by Robertson et al. (2010) might not be able to correctly evaluate the economic value of some feasible solutions.

6.8 Larger Refinery Problem

In this section, the proposed Lagrangian decomposition approach is applied on a larger refinery example and compared to standard MINLP solvers and the basic sequential procedure presented in section 6.7. The refinery planning problem is based on a crude distillation simulation model developed by Gueddar and Dua (2010). This crude distillation model is based on a layered artificial neural network (ANN). This ANN is generated by solving an MINLP which aims at fitting empirical data for atmospheric distillation of several crudes (crude assays) while simplifying the calculations. The model obtained is able to predict cut yields and cut properties from the chosen cut points and properties of the inlet crude. Figure 6.16 depicts the full nonlinear planning model. Three different types of crude blends are processed in three different CDU operating modes and five crude cuts are produced: LPG (liquefied petroleum gas), naphta, kerosene, diesel, and residue. Each discrete CDU mode is defined by the distillation cut point between diesel and residue fractions: 340, 360, or 380°C. The decision variables for each CDU mode are composed of individual crude flows and distillation cut points between naphta, kerosene and diesel fractions. The three streams produced for each fraction are then blended into cut pools. The bilinear pooling constraints discussed in section 6.2 are classically used to calculate the properties of each cut pool. Table 6.5 shows market prices for each distillation cut and corresponding property specifications. Crude availability constraints are also included in the model in accordance with the description of the crude-oil scheduling problem. This refinery planning model is composed of 1,831 variables and 1,817 constraints. The full mathematical model is detailled

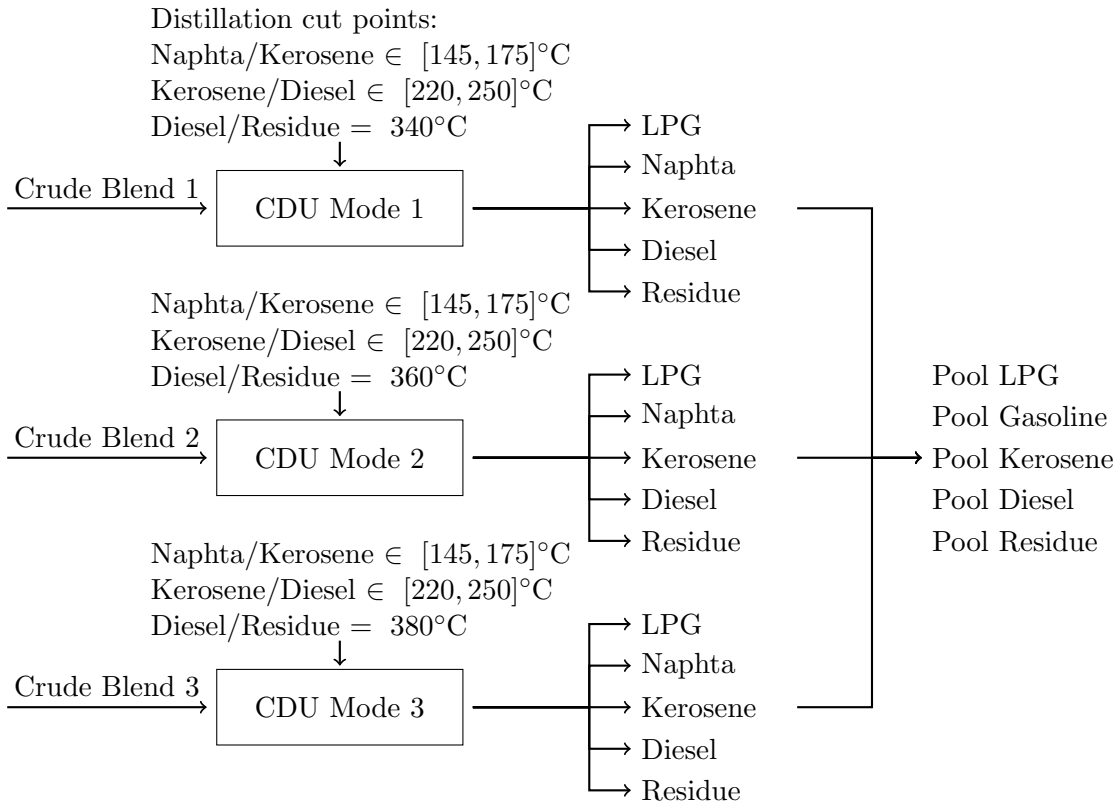


Figure 6.16: Planning model for larger refinery problem.

in appendix D.

The crude-oil scheduling problem is based on example 3 from Lee et al. (1996) (see Figure 6.17) with the parameters described in Table 6.6. It consists of three crude arrivals, three storage tanks, three charging tanks (one for each type of crude mixture), and two identical CDUs whose respective scheduled feedstocks are merged when linked to the refinery planning problem. Seven different crudes are available, and fourteen transfer operations can be executed to prepare the different crude blends. When 6 priority-slots are used, the crude-oil scheduling problem is composed of 1,814 variables (84 binary) and 2,338 constraints.

Table 6.7 and Figure 6.18 show the iteration statistics for the Lagrangian decomposition method using 6 priority-slots for the crude-oil scheduling model. The maximum step size parameter $\overline{\alpha}$ is set to 1 and the step bound parameter $\overline{\delta}$ to 0.05. Because we were not

Table 6.5: Crude cut prices and specification for larger refinery problem.

Crude Cut	Price	Specifications
LPG	8.5	None
Naphta	8.0	specific gravity $\in [0.72, 0.775]$ motor octane number ≥ 45 research octane number ≥ 45 sulfur weight content $\leq$ 120ppm
Kerosene	7.0	specific gravity $\in [0.775, 0.84]$ freeze point $\leq -40°$C
Diesel	8.0	specific gravity $\in [0.82, 0.86]$ cetane number ≥ 48 cloud point $\leq 4°$C sulfur weight content $\leq$ 2800ppm
Residue	6.5	None

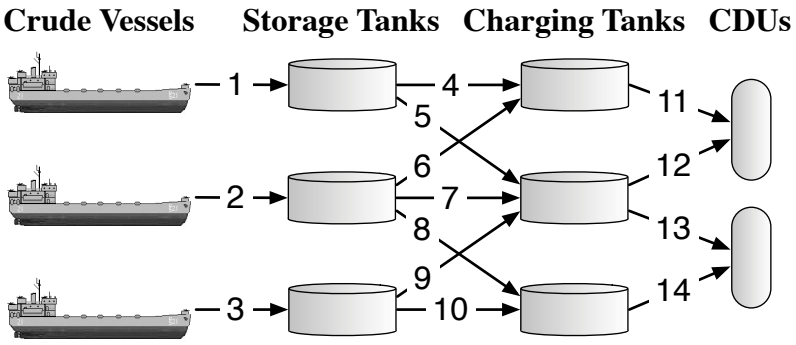


Figure 6.17: Refinery crude-oil scheduling system for COSP3.

able to solve the refinery planning problem to global optimality, we used CONOPT as the NLP solver. Other local NLP solvers as SNOPT and IPOPT did not perform as well (slow convergence, poor solutions or local infeasibilities). As the refinery planning problem is not solved to global optimality, it is not possible to rigorously estimate global optimality for the solution obtained for the full-space problem. Therefore, the convergence tolerance ε is set to 0.01 instead of 0.0001 as in the case study. Furthermore, many iterations are needed to generate enough cutting planes and make the pure restricted dual problem bounded. In order to achieve convergence after a few iterations, the Lagrangian gap is calculated using

Table 6.6: Crude-oil scheduling data for larger refinery problem.

Scheduling horizon			12 days
Vessels	Arrival time	Composition	Amount of crude
Vessel 1	0	100% A	500
Vessel 2	4	100% B	500
Vessel 3	8	100% C	500
Storage tanks	Capacity	Initial composition	Initial amount of crude
Tank 1	[0, 1,000]	100% D	200
Tank 2	[0, 1,000]	100% E	200
Tank 3	[0, 1,000]	100% F	200
Charging tanks	Capacity	Initial composition	Initial amount of crude
Tank 1 (mix 1)	[0, 1,000]	100% G	300
Tank 2 (mix 2)	[0, 1,000]	100% E	500
Tank 3 (mix 3)	[0, 1,000]	100% F	300
Crudes	Property 1 (sulfur concentration)		Crude unit cost
Crude A	0.01		6
Crude B	0.085		6.5
Crude C	0.06		5.5
Crude D	0.02		7.2
Crude E	0.05		6.7
Crude F	0.08		6.2
Crude G	0.03		7.5
Crude mixtures	Property 1 (sulfur concentration)		Maximum number of batches
Crude mix 1	[0.025, 0.035]		6
Crude mix 2	[0.045, 0.065]		6
Crude mix 3	[0.075, 0.085]		6
Unloading flowrate	[0, 50]	Transfer flowrate	[0, 50]
Distillation flowrate	[5, 50]	Minimum duration of distillations	1 day

the objective value of the hybrid dual problem, which is always bounded, as follows:

$$\frac{v\left(\tilde{P}_R(\lambda^K)\right) - v\left(\hat{P}_D^K\right)}{v\left(\tilde{P}_R(\lambda^K)\right)} \stackrel{?}{\leq} \varepsilon$$

The proposed approach converges on the Lagrangian gap in 15 iterations. The final dual gap, although it does not represent a valid optimality gap, is 3.8%. 83% of the time is spent on solving the crude-oil scheduling MILPs. The optimal value of the Lagrange multiplier is given in Table 6.8. With 7 priority-slots, the decomposition procedure converges in 18 iterations and 4,451 seconds, the increase of CPU time being explained mostly by the increased number of priority-slots. The solution obtained is slightly lower than the previous one because the algorithm did not fully converge within tight tolerances.

Table 6.9 presents computational performances of the different approaches. Clearly, the

Table 6.7: Lagrangian iterations statistics for larger refinery problem (6 priority-slots, NLP=CONOPT).

Iteration	Pure Dual $v\left(P_D^K\right)$	Hybrid Dual $v\left(\hat{P}_D^K\right)$	Step Size α^K	Modified Relaxation $v\left(\tilde{P}_R(\lambda^K)\right)$	Modified Heuristic $v\left(\tilde{P}_H(y_B^K)\right)$	Original Heuristic $v\left(P_H(y_B^K)\right)$	CPU Time
1	—	—	—	266.226	—	—	43s
2	—	-2.166	1	417.311	257.943	222.339	102s
3	—	253.546	1	411.657	258.488	245.306	173s
4	—	281.076	0.857	325.906	258.106	—	235s
5	—	260.746	1	288.587	258.104	—	298s
6	—	256.301	1	273.197	258.516	245.467	372s
7	—	256.385	1	271.415	258.027	250.989	459s
8	—	258.091	1	264.974	257.961	233.713	539s
9	—	258.847	0.861	265.149	257.961	233.713	610s
10	—	257.623	1	267.541	257.740	228.230	682s
11	—	259.500	0.868	261.237	257.750	247.971	769s
12	—	259.170	1	261.512	257.750	247.971	850s
13	—	259.126	0.759	264.109	258.027	250.989	913s
14	—	259.121	1	263.708	258.306	—	979s
15	—	259.592	0.774	260.857	258.516	238.763	1,045s

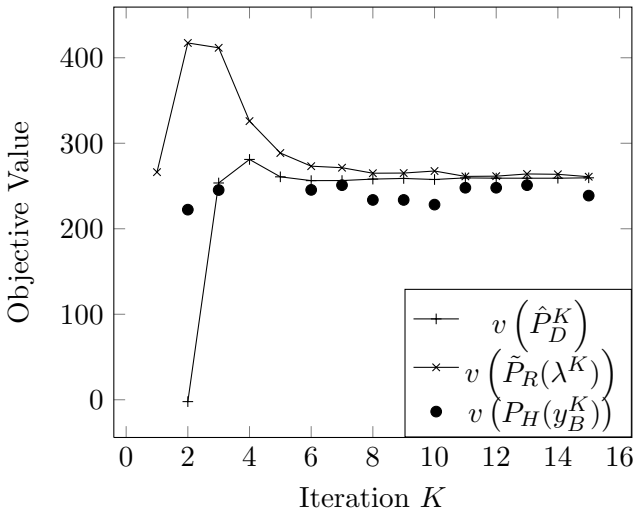


Figure 6.18: Lagrangian iteration objective values for larger refinery problem (6 priority-slots, NLP=CONOPT).

Table 6.8: Optimal Lagrange multipliers for each crude and each CDU mode.

Crude	CDU Mode	Initial value (=Price)	Optimal value
A	1	6.0	6.169
	2	6.0	6.207
	3	6.0	7.695
B	1	6.5	6.971
	2	6.5	6.987
	3	6.5	7.026
C	1	5.5	6.062
	2	5.5	6.009
	3	5.5	6.054
D	1	7.2	7.186
	2	7.2	7.262
	3	7.2	7.962
E	1	6.7	6.859
	2	6.7	6.878
	3	6.7	6.898
F	1	6.2	6.340
	2	6.2	6.226
	3	6.2	6.293
G	1	7.5	6.930
	2	7.5	7.360
	3	7.5	7.360

Table 6.9: Comparative performance of several MINLP algorithms for larger refinery problem (NLP solver: CONOPT).

	6 priority-slots		7 priority-slots	
MINLP Solver	Objective Value	CPU Time	Objective Value	CPU Time
Proposed	250.989	1,045s	250.128	4,451s
Sequential	116.814	46s	—	79s
DICOPT	—	+3,600s	—	+3,600s
AlphaECP	—	+3,600s	—	+3,600s
SBB	—	+3,600s	—	+3,600s

Table 6.10: Blend compositions in the optimal solution of larger refinery problem.

Crude	Blend 1	Blend 2	Blend 3
A	0.164	49.480	
B		33.293	16.707
C		50.000	
D	0.066	19.792	
E	3.500	55.943	2.982
F			50.000
G	9.480		

Lagrangian decomposition procedure is much more robust than the other algorithms. Only the sequential approach was able to deliver a solution with 6 priority-slots, but its objective value is much lower (50% reduction) than the best known solution. Indeed, the scheduling solution obtain during the first stage of the sequential procedure does not take into account the economic impact on the refinery planning problem, which leads to poor decisions. The standard local MINLP solvers were not able to find a solution within one hour because the MINLP model is too large: 3,645 variables (84 binary) and 4,177 constraints.

Table 6.10 shows the crude compositions of the optimal solution with objective value 250.989. The two CDUs are mostly operated in mode 2 which corresponds to the average cut point for diesel and residue cuts.

6.9 Conclusion

In this chapter, a novel approach towards the integration of planning and scheduling has been developed in the context of oil refining. In particular, a precise crude-oil operations scheduling model and a coarse refinery planning model were optimized simultaneously using Lagrangian decomposition. It makes use of Lagrange multipliers as a way to communicate economic information between the two subsystems. The methodology leads to a classical primal-dual iterative algorithm to solve the Lagrangian dual problem. The critical multiplier update step is performed by solving a new hybrid restricted dual problem. This approach combines the strengths of cutting planes and subgradient steps and does not require to

define heuristic updates of parameters during iterations.

Although it is not guaranteed, our results achieved a 0% dual gap for the smaller case study. It is well-known that augmented Lagrangian techniques (see Li and Ierapetritou, 2010) can ensure closing the dual gap for any instance. However, this would require to solve the refinery planning subproblem to global optimality, which is not yet achievable in an industrial context. It is therefore more practical to use standard duality in order to obtain feasible heuristic solutions for the integrated problem.

The proposed Lagrangian decomposition procedure has been applied to a larger and more complex refinery problem. The refinery planning problem is based on a crude distillation simulation model that has been developed by Gueddar and Dua (2010) using an artificial neural network to fit empirical data. The crude-oil scheduling problem is the third example of Lee et al. (1996). The results have shown that the proposed approach remains very competitive compared to other MINLP algorithms, namely the two-step MILP-NLP sequential procedure and standard MINLP solvers.

Chapter 7

Conclusion

In this thesis we have developed models and solution methods to address the problem of short-term scheduling of refinery crude-oil operations. In chapter 2, we proposed a unified modeling approach and solution methods for solving process scheduling problems and applied it to single-stage and multi-stage batch scheduling problems. In chapter 3, we applied this approach to the crude-oil operations scheduling problem and introduced a two-step MILP-NLP procedure to solve the MINLP model. In chapter 4, we developed a symmetry-breaking approach based on regular languages for the SOS time representation in order to improve its computational performance. In chapter 5, we integrated CP bound contraction techniques within CPLEX in order to tighten the linear relaxation of the SOS MINLP model during the branch & bound search. Finally, in chapter 6 we used a Lagrangian decomposition approach to integrate the refinery planning problem with the crude-oil operations scheduling problem, and we introduced a new hybrid dual algorithm for determining the optimal Lagrange multipliers.

7.1 Time Representations and Mathematical Models for Process Scheduling Problems

In chapter 2, we presented a unified modeling approach for process scheduling problems based on four different time representations. All time representations make use of the generic concepts of *priority-slots* and *operations*. We established theoretical relations between these time representations and showed that the MOS time representation can lead to the most compact mathematical formulations as it requires fewer priority-slots.

Using common sets of parameters, variables, and constraints, we developed MILP models for each time representation. These models only differ by a few representation-specific constraints, namely constraints (2.12)-(2.16). The proposed models were automatically strengthened using the non-overlapping graph of the scheduling system. Strengthened constraints (2.17)-(2.21) and (2.23)-(2.26) were obtained using the maximal cliques and maximal bicliques of the graph.

A typical issue with scheduling problems is to select the number of priority-slots to be used. This issue can be avoided in specific cases as precedence-based scheduling model for batch processes with fixed number of batches. In this chapter, we presented generic solution methods for selecting the number of priority-slots. The performance of the additive approach has been improved by using cutoffs and the new constraint (2.27), which reduces the search space at each iteration.

The MOS, MOS-SST and MOS-FST time representations have been applied to single-stage batch scheduling problems with up to 29 orders processed on 4 parallel units. Using the available data for unit processing times, we derived tight cardinality bounds for the number of orders to be processed on each unit. In the case of the MOS mode, we presented a symmetry-breaking constraint (2.29) that was used with unit cardinality bounds in order to derive the problem-specific strengthened constraints (2.30)-(2.32). The computational results show that all instances can be solved to rigorous global optimality in less than 810s, the worst case being the instance with 25 orders. Furthermore, the results showed that during the additive approach, the MOS model for instance SSBSP18 with 10 priority-slots was solved in less than 2s with constraint (2.27) while the same iteration takes more than 100s if it is not used. Similarly, the results showed than the MOS model for instance SSBSP18 with 5 priority-slots was solved in less than 2s with symmetry-breaking constraints (2.29)-(2.32), while it takes more than 100s if they are not used. Finally, the results showed that the strengthened non-overlapping constraint (2.20) can lead to a decrease of CPU times from 3,288s to less than 2s for the same model. The MOS-SST approach failed to solve instances with 18, 25 and 29 orders in less than 1,000s. The MOS-FST approach only failed

to solve the instance with 29 orders in less than 1,000s.

Additionally, the MOS, MOS-SST and MOS-FST time representations have been applied to multi-stage batch problems involving up to 10 orders, 5 stages and 25 units. The MOS approach was able to solve instances with 5 and 8 orders to global optimality in less than 1,000s and failed on instance MSBSP10. No instance was solved to global optimality with the MOS-SST and MOS-FST approaches in the given time limit.

The proposed unified modeling approach can be seen an integration of several approaches studied over the past. The main challenge was to use identical concepts in order to derive efficiently strengthened mathematical formulations for all time representations that can be applied to a large variety of problems. The SOS time representation has been introduced in this work and needs to be further developed in order to efficiently solve scheduling problems. In chapter 4, we studied the application of this time representation in the context of crude-oil scheduling problems. The main limitation of the proposed approach is its inability to handle discrete resource constraints, or inventory transfers at start and/or end times of process operations. In order to apply this approach to a larger class of problems including these types of constraints (for example in STN and RTN scheduling problems), the proposed time representations need to be extended to sequence start and end times of all operations independently.

On the algorithmic part, the additive approach, although briefly cited in previous papers (for example, see Maravelias and Grossmann, 2003), has been enhanced with the use of cutoffs and constraint (2.27). To the best of our knowledge, the proposed multiplicative approach has not been applied to discrete-time scheduling models in the past. It benefits from the fact that any solution obtained at a given iteration is feasible for the next iteration, which justifies objective cutoffs as well as stopping criteria based on objective improvement tolerances. As the initial number of priority-slots may have a large impact on the global solution time, this approach can benefit from the work of Maravelias and Papalamprou (2009) who proposed a strategy for the selection of the length of each time period based on polyhedral results for the discrete-time MILP.

7.2 Short-Term Scheduling of Crude-Oil Operations

In chapter 3, we adapted the MOS, MOS-SST, and MOS-FST models presented in chapter 2 to the crude-oil scheduling problem. We showed how the non-overlapping graph can be used to derive non-trivial strengthened constraints and we introduced new general symmetry-breaking constraints for the MOS time representation.

We presented a simple two-step MILP-NLP decomposition procedure that provides an estimate of optimality gap for the returned solution. The first stage MILP is solved using the additive approach for MOS and MOS-SST time representations and the multiplicative approach for the MOS-FST time representation. Since the stopping criterion that is used (no improvement of the objective value) cannot guarantee global optimality of the solution returned for the MILP, the upper bound and therefore the estimate of optimality gap are not rigorous. In practice, the upper bound was always valid except in the case of the MOS-SST model applied to instance COSP2 (the upper bound found was 101.174 instead of 101.175). All instances were solved within a 3% optimality gap using the MOS time representation. An important challenge is to reduce this optimality gap for problems displaying stronger non-convexities. This issue has been addressed in chapter 5 for minimizing total logistics costs in the crude-oil operations scheduling problem.

In terms of computational performance, the proposed MINLP procedure behaves generally better than other MINLP solvers on the crude-oil scheduling problem. The approach benefits from the fact that it is possible to derive a simple, yet very tight MILP relaxation of the global MINLP. It is interesting to notice that DICOPT displays very similar results. Indeed, the first iteration of the outer-approximation algorithm implemented in DICOPT is very similar to the proposed two-step procedure. The only difference comes from the fact that in DICOPT, the MILP contains first-order Taylor expansion of the nonlinear constraints around the solution of the relaxed MINLP (i.e. the NLP relaxation). However, the issue of selecting the optimal number of priority-slot was not address while using DICOPT.

The results show that the MILP model was always solved within 20s for the MOS time

representation and within 400s for the MOS-SST time representation. In the MOS-FST time representation, only the first instance COSP1 was solved within 1,000s. The poor performance of this discrete-time formulation is due to fact that constraint (2.24) does not handle effectively variable processing times. Instead, it is preferable to use the discrete-time formulation proposed in Lee et al. (1996). This type of discrete-time representation differs conceptually from the representations proposed in this thesis since one execution of a given operation may be split into adjacent smaller executions, one per time period. This type of formulation has many advantages when processing times are variables. However, it may require additional binary variables to represent the priority-slots corresponding to the actual start and end times of each operation executions (before splitting). For example, this is required when the scheduling problem involves cardinality constraints.

The solutions obtained for the problem COSP1 showed that schedule optimization can lead up to a 13.2% profit increase over manual or heuristic scheduling approaches. The impact of uncertainty in terms of vessel arrival has been studied in the case of problem COSP2 showing that a reactive scheduling approach may be sufficient in the case of profit maximization. The incentive of using advanced strategies for handling uncertainties (e.g. stochastic programming or robust optimization) has not been precisely evaluated.

7.3 Single-Operation Sequencing Model for Crude-Oil Operations Scheduling

In chapter 4, we studied the SOS time representation applied to the crude-oil operations scheduling problem. As the non-overlapping graph can be used to generate many non-redundant strengthened non-overlapping constraints, we developed three heuristic selection rules for maximal cliques and bicliques that aim at reducing the number of such constraints.

It was shown that the SOS model may display many symmetric solutions. Therefore, we developed symmetry-breaking constraints in order to reduce the size of the feasible space of the MILP. The symmetry-breaking approach is based on a problem-specific sequencing

rule that needs to be derived by the modeler taking into account the scheduling constraints involved in the system. We proposed to use regular languages and deterministic finite automata to represent, store and model this sequencing rule. Additionally, we determined the maximum number of operations needed to represent any solution for each instance. The SOS model has been solved using the direct approach and fixing the number of priority-slots to the maximum number of operations.

The four crude-oil scheduling instances were all solved in less than 310s using the SOS time representation, showing that it is less efficient than the MOS time representation. This is mostly due to the large size of the corresponding MILP. On average, the SOS model contains 329% more binary variables, 1188% more continuous variables, and 275% more constraints than the MOS model. Although global optimality cannot be rigorously achieved with the proposed approach, the MILP solution returned by the SOS model is always optimal. Finally, we have shown that the symmetry-breaking sequencing rule reduces the CPU time for solving problem COSP1 with 13 priority-slots from more than 1 hour to only 2s.

The comparative study showed that the SOS time representation is less promising than the MOS time representation due to its larger model size. However, sequencing rules can be used to enforce complex sequencing constraints instead of symmetry-breaking constraints. In principle, they could also be applied to time representations other than SOS even though multiple operations may be assigned to each priority-slot.

7.4 Tightening the Linear Relaxation of a Crude-Oil Operations Scheduling MINLP Using CP

In chapter 5, we developed an integrated MILP / CP algorithm in order to tighten the MILP relaxation of a crude-oil scheduling problem modeled with the SOS time representation. The objective function studied corresponds to the minimization of the logistics cost and involves bilinear terms due to storage costs. As shown in Lee et al. (1996), this objective function can be artificially linearized using a discrete-time representation. However, it is an

interesting case study for testing and enhancing the MILP-NLP decomposition approach. In particular, the optimality gaps obtained with the basic two-step procedure range from 11.7% to 21.9%. Therefore there is a clear incentive to improve the upper bound obtained at the MILP stage.

The hybrid search algorithm is based on deriving variable bounds for volume and time variables in order to generate valid McCormick convex envelopes during the search. CPLEX is used as the main search algorithm, thus benefiting from its advanced branching strategies, heuristics, cutting planes, and efficient memory management. ILOG CP is used to update variable bounds at each node and tightening McCormick cuts are generated. The optimality gap for all instances was reduced to less than 7% for all instances. Additionally, a new improved solution was obtained for problem COSP2. The comparison with other MINLP solvers show that this approach outperforms other MINLP algorithms, both in terms of CPU times and optimality of the solutions returned. As demonstrated in chapter 3, DICOPT is the best MINLP alternative.

The proposed MILP / CP approach can be applied to any MINLP with bilinear terms, or in general any MINLP with nonlinear terms that can be linearly relaxed using variable bounds. The model used for CP constraint propagation is critical in the determination of tight bounds at each node. In this work, the mathematical formulation has been used directly as a CP model that proved to be sufficient in the case of the crude-oil operations scheduling problem. Depending on the structure of the problem, the CP model may have to be further enhanced. Also, the CP model can be used to perform logic inference in addition to the classic LP relaxation. Finally, the approach would benefit from spatial branch & bound techniques in order to further improve the upper bound and potentially select better sequences of operations for the NLP stage. The work developed in this chapter is closely related to the solvers SIMPL (see Yunes et al., 2010) and SCIP (for linear problems only, see Achterberg, 2004).

7.5 Integration of Refinery Planning and Crude-Oil Scheduling using Lagrangian Decomposition

In chapter 6, we developed a Lagrangian decomposition approach to integrate refinery planning with the crude-oil operations scheduling problem modeled with the MOS time representation. The two subproblems are linked through a single CDU feedstock synchronization constraint, which is dualized for the purpose of the Lagrangian relaxation. The approach benefits from the fact that the full-space MINLP is decomposed into an MILP and an NLP which can be solved effectively by standard solvers. As the approach is based on relaxing the full-space MINLP, the crude-oil scheduling MINLP subproblem is naturally replaced by an MILP relaxation as described in chapters 3, 4 and 5. The Lagrange multipliers are optimized using a hybrid dual problem composed of cutting planes and a subgradient-based boxstep. Furthermore, we presented a simple heuristic procedure for obtaining a lower bound that consists of solving the full-space NLP obtained after fixing the binary variables to the values of the crude-oil scheduling solution obtained at each iteration.

The computational results show that the approach converges quickly in less than 10 iterations (less than 2 min) with 0% optimality gap for a small-scale example solved with 6 and 7 priority-slots. The heuristic procedure found a feasible solution in 58% of the iterations although only 4 different solutions were found. The optimal solution was found in both cases, with any NLP solver for the heuristic, and in reasonable times showing the robustness of the Lagrangian decomposition approach. Among standard MINLP solvers, DICOPT was the best alternative although it did not find the optimal solution in all cases. On a larger example, 15 iterations and 1,045s were needed in order to solve the problem with 6 priority-slots. The main computational challenge is to solve the MILP relaxation of the crude-oil scheduling problem in shorter CPU times. Furthermore, the estimate of optimality gap for the best known solution is 3.8%. However, this estimate is not rigorous since the upper bounds obtained at each iteration are not achieved through global optimization of the relaxed primal problem.

From a practical point of view, the integration of both subproblems may need to be further improved in order to be applied to industrial problems. In section 6.6, we outlined several ideas in order to improve this integration. The first one consists of using a multi-period refinery planning problem in order to consider long-term economic impacts while optimizing short-term scheduling solutions, thus avoiding solving computationally expensive long-term scheduling models. Furthermore, the aggregation of crude-oil feedstock over the scheduling period may lead to inconsistencies as individual CDU batches are not considered in the refinery planning problem. This issue should be addressed by using exactly one period per CDU batch.

7.6 Contributions of the Thesis

The main contributions of the thesis can be summarized as follows.

1. A unified modeling approach for process scheduling problems has been proposed in chapter 2. The methodology can be applied to a large class of scheduling problems including single-stage batch scheduling, multi-stage batch scheduling and crude-oil operations scheduling. It is solely based on the general concepts of priority-slots and operations, which are used to represent any feasible solution by a sequence of multiple or single operations. The corresponding MILP models only differ by a few representation-specific constraints.
2. A new time representation, called single operation sequencing (SOS), has been introduced in chapter 2 and applied to the crude-oil operations scheduling problem in chapters 4 and 5. As the corresponding MILP model displays many symmetric solutions, we proposed to use a symmetry-breaking sequencing rule represented by a deterministic finite automaton and included it in the model as a network flow formulation. Sequencing rules can also be used to represent complex sequencing constraints.
3. A global representation of the non-overlapping constraints in the scheduling problem has been developed in chapter 2. It can be seen as an extension of the disjunc-

tive graph (Balas, 1969) for operations that may be executed several times. The non-overlapping graph has been used to automatically generate strengthened non-overlapping constraints using its maximal cliques and maximal bicliques.

4. A general symmetry-breaking constraint for MOS scheduling models has been introduced in section 3.3.7. As it does not make any assumption on the structure of the scheduling system, it can be applied to any problem.
5. A simple two-step MILP - NLP decomposition procedure has been proposed for solving crude-oil scheduling MINLPs in chapter 3. It takes advantage of the fact that it is possible to derive a tight MILP relaxation for the MINLP. After solving the MILP, the binary variables are fixed and the full-space NLP is solved to obtain a feasible solution with an estimate of the optimality gap.
6. A general integration approach for refinery planning and crude-oil operations scheduling problems has been proposed in chapter 6. It based on a Lagrangian decomposition of the full-space MINLP problem. The decomposability property is obtain by dualizing the linking constraints, which synchronizes CDU feedstocks decisions in both subproblems.
7. We developed a new generic hybrid dual algorithm for optimizing the Lagrange multipliers in the context of Lagrangian decomposition. This approach is based on solving an LP dual problem composed of cutting planes and a subgradient-based boxstep. Only two parameters have to be set up by the user: the maximum step size and the maximum deviation from the subgradient step.

7.7 Recommendations for Future Work

1. As mentioned in chapter 2, a major limitation of the proposed time representations is that operations must be sequenced as a whole. However, in network-based scheduling problems, reaction operations are often decomposed in 3 phases: reactor charging (considered as instantaneous), reaction process, reactor discharging (also considered

as instantaneous). In order to track inventory levels, it is necessary to sequence start and end events, *independently*. This can be accomplished by extending to proposed time representations and assigning simultaneously two priority-slots to each operation execution. Any solution can then be represented by a sequence of start and end events, corresponding to all transfer events. Additionally, this concept can be used to model discrete resource constraint as start and end events correspond to allocation and release events.

2. The additive approach presented in chapter 2 has not been applied to the integrated refinery planning and crude-oil scheduling problem studied in chapter 6. Therefore, the automatic selection of the number of priority-slots has not been addressed yet. The integration approach can be improved by integrating this algorithm to the Lagrangian decomposition procedure. In particular, two different approaches could be investigated. The first one consists of executing the additive approach at each iteration to solve the crude-oil scheduling problem. The second consists of executing the additive approach outside of the primal-dual procedure using the optimal Lagrange multiplier form each iteration as a starting point for the next iteration. Additionally, cutting planes generated at previous iterations may still be re-used for later iteration, thus improving the convergence of the entire algorithm.
3. The two-step MILP-NLP decomposition procedure proposed in chapter 2 can be extended in two different ways. For local optimization, the procedure may iterate by adding outer-approximation cuts generated from the NLP steps (as in DICOPT). The first MILP provides an upper bound, while successive iterations may provide additional feasible solutions, thus improving the lower bound. For global optimization, the NLP problems may be solved with spatial branch and bound and the procedure may iterate by adding Benders cuts to the MILP master problem (as in Benders decomposition). In any case, the algorithm may be adapted to the specific structure of the crude-oil operations scheduling problem.
4. The MILP relaxation tightening approach developed in chapter 5 has not been applied

towards the integration of refinery planning and crude-oil operations scheduling. The results have shown that the best solution for the larger refinery example in chapter 6 has been obtained with an optimality gap of 3.8%. This optimality gap can be reduced by improving the upper bound obtained by the Lagrangian relaxation and therefore by improving the MILP relaxation of the crude-oil scheduling problem as studied in chapter 5.

Chapter 8

Bibliography

Achterberg, T., 2004. SCIP - a framework to integrate constraint and mixed integer programming - a framework to integrate constraint and mixed integer programming. Tech. rep., Zuse Institue Berlin.

Adams, J., Balas, E., Zawack, D., 1988. The shifting bottleneck procedure for job shop scheduling. Management Science 34 (3), 391–401.

Adhya, N., Tawarmalani, M., Sahinidis, N. V., 1999. A Lagrangian approach to the pooling problem. Industrial and Engineering Chemistry Research 38 (5), 1956–1972.

Adjiman, C. S., Androulakis, I. P., Floudas, C. A., 2000. Global optimization of mixed-integer nonlinear problems. AIChE Journal 46 (9), 1769–1797.

Alhajri, I., Elkamel, A., Albahri, T., Douglas, P. L., 2008. A nonlinear programming model for refinery planning and optimisation with rigorous process models and product quality specifications. International Journal of Oil, Gas and Coal Technology 1 (3), 283–307.

Andersen, H. R., Hadzic, T., Hooker, J. N., Tiedermann, P., 2007. A constraint store based on multivalued decision diagrams. In: 13th International Conference on Principles and Practice of Constraint Programming. Vol. 4741 of Lecture Notes in Computer Science. Springer, pp. 118–132.

Audet, C., Brimberg, J., Hansen, P., Le Digabel, S., Mladenović, N., 2004. Pooling problem: Alternate formulations and solution methods. Management Science 50 (6), 761–776.

Audet, C., Hansen, P., Jaumard, B., Savard, G., 2000. A branch and cut algorithm for nonconvex quadratically constrained quadratic programming. Mathematical Programming 87 (1), 131–152.

Balas, E., 1969. Machine sequencing via disjunctive graphs: An implicit enumeration algorithm. Operations Research 17 (6), 941–957.

Balas, E., 1985. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. SIAM Journal on Algebraic and Discrete Methods 6 (3), 466–486.

Baptiste, P., Le Pape, C., Nuijten, W., 2001. Constraint-based Scheduling: Applying Constraint Programming to Scheduling Problems. Vol. 39 of International Series in Operations Research and Management Science. Kluwer Academic Publishers.

Barahona, F., Anbil, R., 2000. The volume algorithm: producing primal solutions with a subgradient method. Mathematical Programming 87 (3), 385–399.

Bazaraa, M. S., Sherali, H. D., 1981. On the choice of step size in subgradient optimization. European Journal of Operational Research 7 (4), 380–388.

Bergamini, M. L., Grossmann, I. E., Scenna, N., Aguire, P., 2008. An improved piecewise outer-approximation algorithm for the global optimization of minlp models involving concave and bilinear terms. Computers and Chemical Engineering 32, 477–493.

Bixby, R. E., Fenelon, M., Gu, Z., Rothberg, E., Wunderling, R., 1999. MIP: Theory and practice - closing the gap. In: System Modelling and Optimization. IFIP Conference Proceedings. pp. 19–50.

Bodington, C. E., Baker, T. E., 1990. A history of mathematical programming in the petroleum industry. Interfaces 20 (4), 117–127.

Camerini, P. M., Fratta, L., Maffioli, F., 1975. On improving relaxation methods by modified gradient techniques. Mathematical Programming Studies 3, 26–34.

Casas-Liza, J., Pinto, J. M., 2005. Optimal scheduling of a lube oil and paraffin production plant. Computers and Chemical Engineering 29 (6), 1329–1344.

Castro, P. M., Grossmann, I. E., 2005. New continuous-time milp model for the short-term scheduling of multistage batch plants. Industrial and Engineering Chemistry Research 44 (24), 9175–9190.

Castro, P. M., Grossmann, I. E., 2006. An efficient milp model for the short-term scheduling of single stage batch plants. Computers and Chemical Engineering 30 (6-7), 1003–1018.

Castro, P. M., Grossmann, I. E., Novais, A. Q., 2006. Two new continuous-time models for the scheduling of multistage batch plants with sequence dependent changeovers. Industrial and Engineering Chemistry Research 45 (18), 6210–6226.

Cerdá, J., Henning, G. P., Grossmann, I. E., 1997. A mixed-integer linear programming model for short-term scheduling of single-stage multiproduct batch plants with parallel lines. Industrial and Engineering Chemistry Research 36 (5), 1695–1707.

Cheney, E. W., Goldstein, A. A., 1959. Newton's method for convex programming and Tchebycheff approximation. Numerische Mathematik 1 (1), 253–268.

Côté, M. C., Gendron, B., Rousseau, L. M., 2007. Modeling the regular constraint with integer programming. In: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Vol. 4510 of Lecture Notes in Computer Science. Springer, pp. 29–43.

Dua, V., 2010. A mixed-integer programming approach for optimal configuration of artificial neural networks. Chemical Engineering Research and Design 88 (1), 55–60.

Duran, M. A., Grossmann, I. E., 1986. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Mathematical Programming 36 (3), 307–339.

Erdirik-Dogan, M. E., Grossmann, I. E., 2008. Simultaneous planning and scheduling of single-stage multi-product continuous plants with parallel lines. Computers and Chemical Engineering 32 (11), 2664–2683.

Fisher, M. L., 1981. The Lagrangian relaxation method for solving integer programming problems. Management Science 27 (1), 1–18.

Fisher, M. L., 1985. An application oriented guide to Lagrangian relaxation. Interfaces 15, 10–21.

Floudas, C. A., Aggarwal, A., 1990. A decomposition strategy for global optimum search in the pooling problem. ORSA Journal on Computing 2 (3), 225–235.

Floudas, C. A., Lin, X., 2004. Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review. Computers and Chemical Engineering 28, 2109–2129.

Floudas, C. A., Visweswaran, V., 1993. A primal-relaxed dual global optimization approach. Journal of Optimization Theory and Applications 78 (2), 187–225.

Foulds, L. R., Haugland, D., Jörnsten, K., 1992. A bilinear approach to the pooling problem. Optimization 24 (1 & 2), 165–180.

Frangioni, A., 2005. About Lagrangian methods in integer optimization. Annals of Operations Research 139 (1), 163–193.

Furman, K. C., Jia, Z., Ierapetritou, M. G., 2007. A robust event-based continuous time formulation for tank transfer scheduling. Industrial and Engineering Chemistry Research 46 (26), 9126–9136.

Garvin, W. W., Crandall, H. W., John, J. B., Spellman, R. A., 1957. Applications of linear programming in the oil industry. Management Science 3 (4), 407–430.

Goffin, J. L., Haurie, A., Vial, J. P., 1992. Decomposition and nondifferentiable optimization with the projective algorithm. Management Science 38 (2), 284–302.

Goltz, H.-J., Matzke, D., 1998. Practical Aspects of Declarative Languages. Vol. 1551 of Lecture Notes in Computer Science. Springer, Ch. University Timetabling Using Constraint Logic Programming, pp. 320–334.

Gomory, R. E., 1958. Outline of an algorithm for integer solutions to linear programs. Bulletin of the American Mathematical Society 64 (5), 275–278.

Grossmann, I. E., 2002. Review of nonlinear mixed-integer and disjunctive programming techniques. Optimization and Engineering 3 (3), 227–252.

Grossmann, I. E., 2005. Enterprise-wide optimization: A new frontier in process systems engineering. AIChE Journal 51 (7), 1846–1857.

Gueddar, T., Dua, V., 2010. Novel model reduction techniques for refinery-wide energy optimization. Applied Energy Journal.

Guignard, M., 2003. Lagrangean relaxation. Top 11 (2), 151–228.

Gupta, S., Karimi, I. A., 2003. An improved milp formulation for scheduling multiproduct multistage batch plants. Industrial and Engineering Chemistry Research 42, 2365–2380.

Hart, W. D., 1978. L.P. behavior - recursion example comments. ACM SIGMAP Bulletin 25, 29–33.

Haverly, C. A., 1978. Studies of the behavior of recursion for the pooling problem. ACM SIGMAP Bulletin 25, 19–28.

Haverly, C. A., 1980. Recursion model behavior: more studies. ACM SIGMAP Bulletin 28, 39–41.

Held, M., Karp, R. M., 1971. The traveling-salesman problem and minimum spanning trees: Part ii. Mathematical Programming 1 (1), 6–25.

Held, M., Karp, R. M., 1974. Validation of subgradient optimization. Mathematical Programming 6 (1), 62–88.

Hoda, S., van Hoeve, W.-J., Hooker, J. N., 2010. A systematic approach to mdd-based constraint programming. In: 16th International Conference on Principles and Practice of Constraint Programming. Lecture Notes in Computer Science. Springer.

Hooker, J. N., 2007. Integrated Methods for Optimization. Springer.

Hopcroft, J. E., Ullman, J. D., 1979. Introduction to Automata Theory, Languages and Computation. Addison Wesley.

Hui, C., Gupta, A., van der Meulen, H. A. J., 2000. A novel milp formulation for short-term scheduling of multi-stage multi-product batch plants with sequence-dependent constraints. Computers and Chemical Engineering 24 (12), 2705–2717.

Ierapetritou, M. G., Floudas, C. A., 1998. Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. Industrial and Engineering Chemistry Research 37 (11), 4341–4359.

ILOG Inc., 2007. ILOG CPLEX 11.0 User's Manual.

Janak, S. L., Lin, X., Floudas, C. A., 2004. Enhance continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes: Resource consraints and mixed storage policies. Industrial and Engineering Chemistry Research 43 (10), 2516–2533.

Jia, Z., Ierapetritou, M. G., Kelly, J. D., 2003. Refinery short-term scheduling using continuous time formulation: Crude-oil operations. Industrial and Engineering Chemistry Research 42 (13), 3085–3097.

Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., 2000. Progress in linear programming-based algorithms for integer programming: An exposion. INFORMS Journal on Computing 12 (1), 2–23.

Joly, M., Pinto, J. M., 2003. Mixed-integer programming techniques for the scheduling of fuel oil and asphalt production. Chemical Engineering Research and Design 81 (4), 427–447.

Kallrath, J., 2002. Planning and scheduling in the process industry. OR Spectrum 24 (3), 219–250.

Karuppiah, R., Furman, K. C., Grossmann, I. E., 2008. Global optimization for scheduling refinery crude oil operations. Computers and Chemical Engineering 32 (11), 2745–2766.

Kelley, J. E., 1960. The cutting-plane method for solving convex programs. Journal of the Society for Industrial and Applied Mathematics 8 (4), 703–712.

Kelly, J. D., Mann, J. L., 2003. Crude oil blend scheduling optimization: an application with multimillion dollar benefits. Hydrocarbon Processing 82 (6), 47–53.

Kesavan, P., Allgor, R. J., Gatzke, E. P., Barton, P. I., 2004. Outer approximation algorithms for separable nonconvex mixed-integer nonlinear programs. Mathematical Programming 100 (3), 517–535.

Kondili, E., Pantelides, C. C., Sargent, R. W. H., 1993. A general algorithm for short-term scheduling of batch operations. I: MILP formulation. Computers and Chemical Engineering 17 (2), 211–227.

Ku, H., Karimi, I. A., 1988. Scheduling in serial multiproduct batch processes with finite interstage storage: a mixed integer linear program formulation. Industrial and Engineering Chemistry Research 27, 1840–1848.

Land, A. H., Doig, A. G., 1960. An automatic method of solving discrete programming problems. Econometrica 28 (3), 497–520.

Lasdon, L., Joffe, B., 1990. The relationship between distributive recursion and successive linear programming in refining production planning models. In: NPRA Computer Conference. Seattle, WA.

Lee, H., Pinto, J. M., Grossmann, I. E., Park, S., 1996. Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management. Industrial and Engineering Chemistry Research 35 (5), 1630–1641.

Lemaréchal, C., 1974. An algorithm for minimizing convex functions. In: Proceedings IFIP'74 Congress. pp. 552–556.

Leyffer, S., 2001. Integrating SQP and branch-and-bound for mixed integer nonlinear programming. Computational Optimization and Applications 18 (3), 295–309.

Li, W., Hui, C., Li, A., 2005. Integrating cdu, fcc and product blending models into refinery planning. Computers and Chemical Engineering 29 (9), 2010–2028.

Li, Z., Ierapetritou, M. G., 2010. Production planning and scheduling integration through augmented Lagrangian optimization. Computers and Chemical Engineering 34 (6), 996–1006.

Lin, Y., Schrage, L., 2009. The global solver in the lindo api. Optimization Methods and Software 24 (4-5), 657 – 668.

Liu, Y., Karimi, I. A., 2007. Scheduling multistage, multiproduct batch plants with nonidentical parallel unitsandunlimitedintermediatestorage. Chemical Engineering Science 62 (6), 1549–1566.

Manne, A. S., 1958. A linear programming model of the U. S. petroleum refining industry. Econometrica 26 (1), 67–106.

Maravelias, C. T., Grossmann, I. E., 2003. New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. Industrial and Engineering Chemistry Research 42 (13), 3056–3074.

Maravelias, C. T., Grossmann, I. E., 2006. On the relation of continuous- and discrete-time state-task network formulations. AIChE Journal 52 (2), 843–849.

Maravelias, C. T., Papalamprou, K., 2009. Polyhedral results for discrete-time production planning mip formulations for continuous processes. Computers and Chemical Engineering 33 (11), 1890–1904.

Maravelias, C. T., Sung, C., 2009. Integration of production planning and scheduling: Overview, challenges and opportunities. Computers and Chemical Engineering 33 (12), 1919–1930.

Marchetti, P. A., Cerdá, J., 2009. An approximate mathematical framework for resource-constrained multistage batch scheduling. Chemical Engineering Science 64, 2733–2748.

Margot, F., 2008. Symmetry in integer linear programming. Tepper Working Paper 2008 E-37.

Marsten, R. E., Hogan, W. W., Blankenship, J. W., 1975. The boxstep method for large-scale optimization. Operations Research 23 (3), 389–405.

McCormick, G. P., 1976. Computability of global solutions to factorable nonconvex programs: Part 1 - convex underestimating problems. Mathematical Programming 10, 147–175.

Méndez, C. A., Cerdá, J., 2003. Dynamic scheduling in multiproduct batch plants. Computers and Chemical Engineering 27 (8-9), 1247–1259.

Méndez, C. A., Cerdá, J., 2007. A precedence-based monolithic approach to lotsizing and scheduling of multiproduct batch plants. In: 17th European Symposium on Computer Aided Process Engineering - ESCAPE17. Vol. 24 of Computer Aided Chemical Engineering. pp. 679–684.

Méndez, C. A., Cerdá, J., Grossmann, I. E., Harjunkoski, I., Fahl, M., 2006a. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Computers and Chemical Engineering 30, 913–946.

Méndez, C. A., Grossmann, I. E., Harjunkoski, I., Kaboré, P., 2006b. A simultaneous optimization approach for off-line blending and scheduling of oil-refinery operations. Computers and Chemical Engineering 30, 614–634.

Méndez, C. A., Henning, G. P., Cerdá, J., 2001. An milp continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. Computers and Chemical Engineering 25, 701–711.

Meyer, C. A., Floudas, C. A., 2006. Global optimization of a combinatorially complex generalized pooling problem. AIChE Journal 52 (3), 1027–1037.

Michel, L., van Hentenryck, P., 2003. Comet in context. In: Principles of computing & knowledge: Proceedings of the Paris C. Kanellakis Memorial Workshop. Vol. 41 of ACM International Conference Proceeding Series. pp. 95–107.

Milano, M., Wallace, M., 2006. Integrating operations research in constraint programming. 4OR: A Quarterly Journal of Operations Research 4 (3), 175–219.

Misener, R., Floudas, C. A., 2009. Advances for the pooling problem: Modeling, global optimization, and computational studies. Applied and Computational Mathematics 8 (1), 3–22.

Mitchell, M., 1998. An Introduction to Genetic Algorithms. The MIT Press.

Moro, L. F. L., Pinto, J. M., 2004. Mixed-integer programming approach for short-term crude oil scheduling. Industrial and Engineering Chemistry Research 43, 85–94.

Mouret, S., Grossmann, I. E., Pestiaux, P., 2008. Multi-operations time-slots model for crude-oil operations scheduling. In: 18th European Symposium on Computer Aided Process Engineering - ESCAPE18. Vol. 25 of Computer Aided Chemical Engineering. pp. 593–598.

Mouret, S., Grossmann, I. E., Pestiaux, P., 2009a. A novel priority-slot based continuous-time formulation for crude-oil scheduling problems. Industrial and Engineering Chemistry Research 48 (18), 8515–8528.

Mouret, S., Grossmann, I. E., Pestiaux, P., 2009b. Tightening the linear relaxation of a mixed integer nonlinear program using constraint programming. In: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Vol. 5547 of Lecture Notes in Computer Science. Springer, pp. 208–222.

Mouret, S., Grossmann, I. E., Pestiaux, P., 2010a. Integration of refinery planning and crude-oil scheduling using Lagrangian decomposition. To be submitted.

Mouret, S., Grossmann, I. E., Pestiaux, P., 2010b. Time representations and mathematical models for process scheduling problems. Computers and Chemical Engineering.

Neiro, S. M. S., Pinto, J. M., 2006. Lagrangean decomposition applied to multiperiod planning of petroleum refineries under uncertainty. Latin American Applied Research 36 (4), 213–220.

Nemhauser, G. L., Wolsey, L. A., 1999. Integer and Combinatorial Optimization. Wiley-Interscience.

Pantelides, C. C., 1994. Unified frameworks for optimal process planning and scheduling. In: Rippin, D., Hale, J., Davis, J. (Eds.), Proceedings of the Second International Conference on Foundations of Computer-Aided Process Operations. pp. 253–274.

Pesant, G., 2004. A regular language membership constraint for finite sequences of variables. In: 10th International Conference on Principles and Practice of Constraint Programming. Vol. 3258 of Lecture Notes in Computer Science. Springer, pp. 482–495.

Pham, V., Laird, C. D., El-Halwagi, M., 2009. Convex hull discretization approach to the global optimization of pooling problems. Industrial and Engineering Chemistry Research 48 (4), 1973–1979.

Pinto, J. M., Grossmann, I. E., 1995. A continuous time mixed integer linear programming model for short term scheduling of multistage batch plants. Industrial and Engineering Chemistry Research 34 (9), 3037–3051.

Pinto, J. M., Joly, M., Moro, L. F. L., 2000. Planning and scheduling models for refinery operations. Computers and Chemical Engineering 24, 2259–2276.

Pinto, J. M., Moro, L. F. L., 2000. A planning model for petroleum refineries. Brazilian Journal of Chemical Engineering 17 (4-7), 575–586.

Prasad, P., Maravelias, C. T., 2008. Batch selection, assignment and sequencing in multi-stage multi-product processes. Computers and Chemical Engineering 32 (6), 1106–1119.

Quesada, I., Grossmann, I. E., 1992. An lp/nlp based branch and bound algorithm for minlp optimization. Computers and Chemical Engineering 16, 937–947.

Quesada, I., Grossmann, I. E., 1995a. A global optimization algorithm for linear fractional and bilinear programs. Journal of Global Optimization 6 (1), 39–76.

Quesada, I., Grossmann, I. E., 1995b. Global optimization of bilinear process networks with multicomponent flows. Computers and Chemical Engineering 19 (12), 1219–1242.

Reddy, P. C. P., Karimi, I. A., Srinivasan, R., 2004. A new continuous-time formulation for scheduling crude oil operations. Chemical Engineering Science 59 (6), 1325–1341.

Régin, J.-C., 2003. Global constraints and filtering algorithms. In: Constraints and Integer Programming Combined. Kluwer, Ch. 1.

Robertson, G., Palazoglu, A., Romagnoli, J. A., 2010. Refinery scheduling of crude oil unloading, storing, and processing considering production level cost. In: 20th European Symposium on Computer Aided Process Engineering - ESCAPE20. Vol. 28 of Computer Aided Chemical Engineering. pp. 1159–1164.

Rossi, F., van Beek, P., Walsh, T. (Eds.), 2006. Handbook of Constraint Programming. Elsevier.

Sahinidis, N. V., 1996. Baron: A general purpose global optimization software package. Journal of Global Optimization 8 (2), 201–205.

Sahinidis, N. V., 2003. Global Optimization and Constraint Satisfaction. Vol. 2861 of Lecture Notes in Computer Science. Springer, Ch. The Branch-and-Reduce Approach, pp. 1–16.

Schilling, G., Pantelides, C. C., 1996. A simple continuous-time process scheduling formulation and a novel solution algorithm. Computers and Chemical Engineering 20 (Suppl. 2), S1221–S1226.

Shah, N., 1996. Mathematical programming techniques for crude oil scheduling. Computers and Chemical Engineering 20, 1227–1232.

Shah, N., Pantelides, C. C., Sargent, R. W. H., 1993. A general algorithm for short-term scheduling of batch operations. 2. Computational issues. Computers and Chemical Engineering 17 (2), 229–244.

Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems. In: 4th International Conference on Principles and Practice of Constraint Programming. Vol. 1520 of Lecture Notes in Computer Science. Springer, pp. 417–431.

Tawarmalani, M., Sahinidis, N. V., 2004. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. Mathematical Programming 99 (3), 563–591.

van Hentenryck, P., 1989. Constraint satisfaction in logic programming. The MIT Press.

van Hoeve, W.-J., Pesant, G., Rousseau, L. M., Sabharwal, A., 2009. New filtering algorithms for combinations of among constraints. Constraints 14 (2), 273–292.

Wassick, J. M., 2009. Enterprise-wide optimization in an integrated chemical complex. Computers and Chemical Engineering 33 (12), 1950–1963.

Waterer, H., Johnson, E. L., Nobili, P., Savelsbergh, M. W. P., 2002. The relation of time indexed formulations of single machine scheduling problems to the node packing problem. Mathematical Programming 93 (3), 477–494.

Wenkay, L., Hui, C., Hua, B., Tong, Z., 2002. Scheduling crude oil unloading, storage, and processing. Industrial and Engineering Chemistry Research 41 (26), 6723–6734.

Westerlund, T., Pettersson, F., 1995. An extended cutting plane method for solving convex minlp problems. Computers and Chemical Engineering 19 (Suppl. 1), S131–S136.

Wicaksono, D. S., Karimi, I. A., 2008. Piecewise milp under- and overestimators for global optimization of bilinear programs. AIChE Journal 54 (4), 991–1008.

Wikipedia, 2010. Oil refinery. `http://en.wikipedia.org/wiki/Oil_refinery`.

Yunes, T., Aron, I. D., Hooker, J. N., 2010. An integrated solver for optimization problems. Operations Research 58 (2), 342–356.

Zhang, J., Kim, N., Lasdon, L., 1985. An improved successive linear programming algorithm. Management Science 31 (10), 1312–1331.

Zhang, X., Sargent, R. W. H., 1996. The optimal operation of mixed production facilities: A general formulation and some approaches for the solution. Computers and Chemical Engineering 20 (6-7), 897–904.

Appendices

Appendix A

On Tightness of Strengthened Constraints

We present mathematical results on the tightness of strengthened constraints.

- Constraint (2.17) is at least as tight as constraint (2.4). Indeed, assume constraint (2.17) is satisfied for W' such that $v_1, v_2 \in W'$:

$$Z_{iv_1} + Z_{iv_2} \leq \sum_{v \in W'} Z_{iv} \leq 1$$

- Constraint (2.6) is at least as tight as constraints (2.5). Indeed, assume constraint (2.6) is satisfied for $v_1, v_2 \in W$:

$$\begin{aligned} E_{i_1 v_1} &\leq E_{i_1 v_1} + E_{i_1 v_2} \\ &\leq S_{i_2 v_1} + S_{i_2 v_2} + H \cdot (1 - Z_{i_2 v_1} - Z_{i_2 v_2}) \\ &\leq S_{i_2 v_2} + H \cdot (1 - Z_{i_2 v_2}) + S_{i_2 v_1} - H \cdot Z_{i_2 v_1} \\ &\leq S_{i_2 v_2} + H \cdot (1 - Z_{i_2 v_2}) \end{aligned}$$

- Constraint (2.18) is at least as tight as constraint (2.6). Indeed, assume constraint (2.18) is satisfied for W' such that $v_1, v_2 \in W'$:

$$\begin{aligned} E_{i_1 v_1} + E_{i_1 v_2} &\leq \sum_{v \in W'} E_{i_1 v} \\ &\leq \sum_{v \in W'} S_{i_2 v} + H \cdot \left(1 - \sum_{v \in W'} Z_{i_2 v}\right) \\ &\leq S_{i_2 v_1} + S_{i_2 v_2} + H \cdot (1 - Z_{i_2 v_1} - Z_{i_2 v_2}) + \sum_{v \in W' \setminus \{v_1, v_2\}} (S_{i_2 v} - H \cdot Z_{i_2 v}) \\ &\leq S_{i_2 v_1} + S_{i_2 v_2} + H \cdot (1 - Z_{i_2 v_1} - Z_{i_2 v_2}) \end{aligned}$$

- Constraint (2.19) is at least as tight as constraint (2.18). Indeed, assume constraint (2.19) is satisfied for W':

$$\sum_{v \in W'} E_{i_1 v} \leq \sum_{v \in W'} E_{i_1 v} + \sum_{\substack{i \in T \\ i_1 < i < i_2}} \sum_{v \in W'} D_{iv} \leq \sum_{v \in W'} S_{i_2 v} + H \cdot (1 - \sum_{v \in W'} Z_{i_2 v})$$

- Constraint (2.21a) is at least as tight as constraint (2.13a). Indeed, assume constraint (2.21a) is satisfied for W' such that $v' \in W'$:

$$S_{iv} \leq \sum_{v' \in W'} S_{iv'} \leq t_i$$

- Constraint (2.21b) is at least as tight as constraint (2.13b). Indeed, assume constraint (2.21b) is satisfied for W' such that $v \in W'$:

$$\begin{aligned} S_{iv} &\geq S_{iv} + \sum_{v' \in W' \setminus \{v\}} (S_{iv'} - H \cdot Z_{iv'}) \\ &\geq \sum_{v' \in W'} S_{iv'} - H \cdot \sum_{v' \in W' \setminus \{v\}} Z_{iv'} \\ &\geq t_i - H \cdot (1 - \sum_{v' \in W'} Z_{iv'}) - H \cdot \sum_{v' \in W' \setminus \{v\}} Z_{iv'} \\ &\geq t_i - H \cdot (1 - Z_{iv}) \end{aligned}$$

- Constraint (2.26) is at least as tight as constraint (2.5). Indeed, assume constraint (2.26) is satisfied for sets of operations W_1 and W_2 such that $v_1 \in W_1$ and $v_2 \in W_2$:

$$\begin{aligned} E_{i_1 v_1} &\leq \sum_{v \in W_1} E_{i_1 v} \\ &\leq \sum_{v \in W_2} S_{i_2 v} + H \cdot (1 - \sum_{v \in W_2} Z_{i_2 v}) \\ &\leq S_{i_2 v_2} + H \cdot (1 - Z_{i_2 v_2}) + \sum_{v \in W_2 \setminus \{v_2\}} (S_{i_2 v} - H \cdot Z_{i_2 v}) \\ &\leq S_{i_2 v_2} + H \cdot (1 - Z_{i_2 v_2}) \end{aligned}$$

Appendix B

Crude-Oil Operations Scheduling Examples

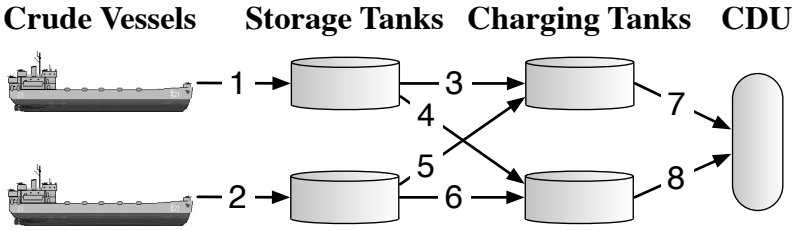


Figure B.1: Refinery crude-oil scheduling system for COSP1.

Table B.1: Data for COSP1.

Scheduling horizon			8 days
Vessels	Arrival time	Composition	Amount of crude (Mbbl)
Vessel 1	0	100% A	1,000
Vessel 2	4	100% B	1,000
Storage tanks	Capacity (Mbbl)	Initial composition	Initial amount of crude (Mbbl)
Tank 1	[0, 1,000]	100% A	250
Tank 2	[0, 1,000]	100% B	750
Charging tanks	Capacity (Mbbl)	Initial composition	Initial amount of crude (Mbbl)
Tank 1 (mix X)	[0, 1,000]	100% C	500
Tank 2 (mix Y)	[0, 1,000]	100% D	500
Crudes	Property 1 (sulfur concentration)		Gross margin ($/bbl)
Crude A	0.01		9
Crude B	0.06		4
Crude C	0.02		8
Crude D	0.05		5
Crude mixtures	Property 1 (sulfur concentration)		Demand (Mbbl)
Crude mix X	[0.015, 0.025]		[1,000, 1,000]
Crude mix Y	[0.045, 0.055]		[1,000, 1,000]
Unloading flowrate	[0, 500]	Transfer flowrate	[0, 500]
Distillation flowrate	[50, 500]	Number of distillations	3

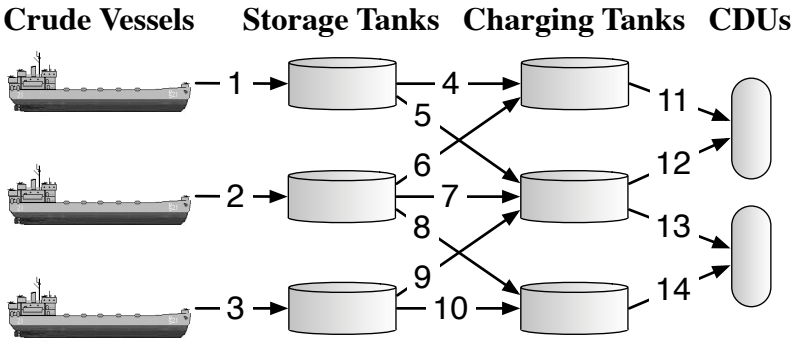


Figure B.2: Refinery crude-oil scheduling system for COSP2 and COSP3.

Table B.2: Data for COSP2.

Scheduling horizon			10 days
Vessels	Arrival time	Composition	Amount of crude (Mbbl)
Vessel 1	0	100% A	1,000
Vessel 2	3	100% B	1,000
Vessel 3	6	100% C	1,000
Storage tanks	Capacity (Mbbl)	Initial composition	Initial amount of crude (Mbbl)
Tank 1	[0, 1,000]	100% A	200
Tank 2	[0, 1,000]	100% B	500
Tank 3	[0, 1,000]	100% C	700
Charging tanks	Capacity (Mbbl)	Initial composition	Initial amount of crude (Mbbl)
Tank 1 (mix X)	[0, 1,000]	100% D	300
Tank 2 (mix Y)	[0, 1,000]	100% E	500
Tank 3 (mix Z)	[0, 1,000]	100% F	300
Crudes	Property 1	Property 2	Gross margin ($/bbl)
Crude A	0.01	0.04	1
Crude B	0.03	0.02	3
Crude C	0.05	0.01	5
Crude D	0.0167	0.333	1.67
Crude E	0.03	0.23	3
Crude F	0.0433	0.133	4.33
Crude mixtures	Property 1	Property 2	Demand (Mbbl)
Crude mix X	[0.01, 0.02]	[0.03, 0.038]	[1,000, 1,000]
Crude mix Y	[0.025, 0.035]	[0.018, 0.027]	[1,000, 1,000]
Crude mix Z	[0.04, 0.048]	[0.01, 0.018]	[1,000, 1,000]
Unloading flowrate	[0, 500]	Transfer flowrate	[0, 500]
Distillation flowrate	[50, 500]	Number of distillations	5

Table B.3: Data for COSP3.

Scheduling horizon			12 days
Vessels	Arrival time	Composition	Amount of crude (Mbbl)
Vessel 1	0	100% A	500
Vessel 2	4	100% B	500
Vessel 3	8	100% C	500
Storage tanks	Capacity (Mbbl)	Initial composition	Initial amount of crude (Mbbl)
Tank 1	[0, 1,000]	100% D	200
Tank 2	[0, 1,000]	100% E	200
Tank 3	[0, 1,000]	100% F	200
Charging tanks	Capacity (Mbbl)	Initial composition	Initial amount of crude (Mbbl)
Tank 1 (mix X)	[0, 1,000]	100% G	300
Tank 2 (mix Y)	[0, 1,000]	100% E	500
Tank 3 (mix Z)	[0, 1,000]	100% F	300
Crudes	Property 1		Gross margin ($/bbl)
Crude A	0.01		1
Crude B	0.085		6
Crude C	0.06		8.5
Crude D	0.02		2
Crude E	0.05		5
Crude F	0.08		8
Crude G	0.03		3
Crude mixtures	Property 1		Demand (Mbbl)
Crude mix X	[0.025, 0.035]		[500, 500]
Crude mix Y	[0.045, 0.065]		[500, 500]
Crude mix Z	[0.075, 0.085]		[500, 500]
Unloading flowrate	[0, 500]	Transfer flowrate	[0, 500]
Distillation flowrate	[50, 500]	Number of distillations	5

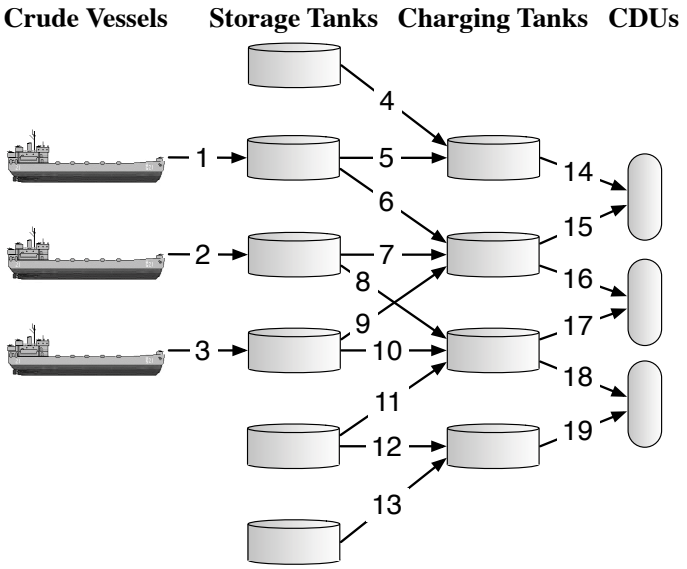


Figure B.3: Refinery crude-oil scheduling system for COSP4.

Table B.4: Data for COSP4.

Scheduling horizon			15 days
Vessels	Arrival time	Composition	Amount of crude (Mbbl)
Vessel 1	0	100% A	600
Vessel 2	5	100% B	600
Vessel 3	10	100% C	600
Storage tanks	Capacity (Mbbl)	Initial composition	Initial amount of crude (Mbbl)
Tank 1	[100, 900]	100% D	600
Tank 2	[100, 1,100]	100% A	100
Tank 3	[100, 1,100]	100% B	500
Tank 4	[100, 1,100]	100% C	400
Tank 5	[100, 900]	100% E	300
Tank 6	[100, 900]	100% E	600
Charging tanks	Capacity (Mbbl)	Initial composition	Initial amount of crude (Mbbl)
Tank 1 (mix X)	[0, 800]	100% F	50
Tank 2 (mix Y)	[0, 800]	100% G	300
Tank 3 (mix Z)	[0, 800]	100% H	300
Tank 4 (mix W)	[0, 800]	100% E	300
Crudes	Property 1		Gross margin ($/bbl)
Crude A	0.03		3
Crude B	0.05		5
Crude C	0.065		6.5
Crude D	0.031		3.1
Crude E	0.075		7.5
Crude F	0.0317		3.17
Crude G	0.0483		4.83
Crude H	0.0633		6.33
Crude mixtures	Property 1		Demand (Mbbl)
Crude mix X	[0.03, 0.035]		[600, 600]
Crude mix Y	[0.043, 0.05]		[600, 600]
Crude mix Z	[0.06, 0.065]		[600, 600]
Crude mix W	[0.071, 0.08]		[600, 600]
Unloading flowrate	[0, 500]	Transfer flowrate	[0, 500]
Distillation flowrate	[20, 500]	Number of distillations	7

Appendix C
Mathematical Models for Crude-Oil Operations Scheduling Problems

C.1 MOS Model

$$\max \quad \sum_{i\in T}\sum_{r\in R_D}\sum_{v\in I_r}\sum_{c\in C} G_c \cdot V_{ivc}$$

s.t. Variable bound and time constraints (3.1)

Cardinality constraints (3.2)

Precedence constraints (3.3)

Continuous distillation constraints (3.4)

Variable constraints (3.5)

Operation constraints (3.6)

Resource constraints (3.8)

Demand constraint (3.9)

Clique-based assignment constraint (3.10)

Clique-based non-overlapping constraint (3.11)

$$Z_{iv} \le \sum_{\substack{v'\in W \\ NO_{vv'}=1}} Z_{(i-1)v'} \qquad i \in T, i \ne 1, v \in W$$

$$\sum_{v\in W} Z_{iv} \ge 1 \qquad i \in T$$

$$S_{iv}, D_{iv}, E_{iv}, V^t_{iv}, V_{ivc}, L^t_{ir}, L_{irc} \ge 0 \qquad i \in T, v \in W, c \in C, r \in R$$

$$Z_{iv} \in \{0,1\} \qquad i \in T, v \in W$$

C.2 MOS-SST Model

$$\max \quad \sum_{i\in T}\sum_{r\in R_D}\sum_{v\in I_r}\sum_{c\in C} G_c \cdot V_{ivc}$$

s.t. Variable bound and time constraints (3.1)

Cardinality constraints (3.2)

Precedence constraints (3.3)

Continuous distillation constraints (3.4)

Variable constraints (3.5)

Operation constraints (3.6)

Resource constraints (3.8)

Demand constraint (3.9)

Clique-based assignment constraint (3.10)

Clique-based non-overlapping constraint (3.11)

$$t_{i-1} \le t_i \qquad i \in T, i \neq 1$$

$$\sum_{v\in W'} S_{iv} \le t_i \qquad i \in T, W' \in clique(G_{NO})$$

$$\sum_{v\in W'} S_{iv} \ge t_i - H \cdot (1 - \sum_{v\in W'} Z_{iv}) \qquad i \in T, W' \in clique(G_{NO})$$

$$\sum_{v\in W} Z_{iv} \ge 1 \qquad i \in T$$

$$S_{iv}, D_{iv}, E_{iv}, V^t_{iv}, V_{ivc}, L^t_{ir}, L_{irc} \ge 0 \qquad i \in T, v \in W, c \in C, r \in R$$

$$Z_{iv} \in \{0,1\} \qquad i \in T, v \in W$$

$$t_i \in [0, H] \qquad i \in T$$

C.3 MOS-FST Model

$$\max \quad \sum_{i\in T}\sum_{r\in R_D}\sum_{v\in I_r}\sum_{c\in C} G_c \cdot V_{ivc}$$

s.t. Variable bound and time constraints (3.1)

Cardinality constraints (3.2)

Precedence constraints (3.3)

Continuous distillation constraints (3.4)

Variable constraints (3.5)

Operation constraints (3.6)

Resource constraints (3.8)

Demand constraint (3.9)

Clique-based assignment constraint (3.10)

Clique-based non-overlapping constraint (3.11)

$$S_{iv} = t_i \cdot Z_{iv} \qquad i \in T, v \in W$$

$$S_{iv}, D_{iv}, E_{iv}, V^t_{iv}, V_{ivc}, L^t_{ir}, L_{irc} \geq 0 \qquad i \in T, v \in W, c \in C, r \in R$$

$$Z_{iv} \in \{0, 1\} \qquad i \in T, v \in W$$

$$t_i = \frac{i-1}{n} \cdot H \qquad i \in T$$

C.4 SOS Model

$$\max \quad \sum_{i\in T}\sum_{r\in R_D}\sum_{v\in I_r}\sum_{c\in C} G_c \cdot V_{ivc}$$

s.t. Variable bound and time constraints (3.1)

Cardinality constraints (3.2)

Precedence constraints (3.3)

Continuous distillation constraints (3.4)

Variable constraints (3.5)

Operation constraints (3.6)

Resource constraints (3.8)

Demand constraint (3.9)

Clique-based assignment constraint (3.10)

Clique-based non-overlapping constraints (4.1)

Biclique-based non-overlapping constraints (4.2)

Symmetry-breaking constraints (4.3)

$$\sum_{v\in W} Z_{iv} \geq 1 \qquad i \in T$$

$$S_{iv}, D_{iv}, E_{iv}, V^t_{iv}, V_{ivc}, L^t_{ir}, L_{irc} \geq 0 \qquad i \in T, v \in W, c \in C, r \in R$$

$$Z_{iv} \in \{0,1\} \qquad i \in T, v \in W$$

Appendix D

Mathematical Model for the Refinery Planning Problem

In this section, the ANN model developed in Gueddar and Dua (2010) for CDU simulations is presented. It is based on a layered directed graph which represents the model calculations (see Fig. D.1). Each node in the input/output layers correspond to one input/output variable. Each node $j = 1, \ldots, N_n$ in the intermediate layer $l = 1, \ldots, N_l$ corresponds to an *activation variable* a_j^l and a *transformed variable* h_j^l. The activation variables are calculated from the transformed variables of the previous layer using an affine expression while the transformed variables are calculated by applying the hyperbolic tangent to their associated activation variable. The ANN equations are expressed as follows.

$$a_j^1 = \sum_{i=1}^{N_x} w_{ji}^1 x_i + b_j^1 \qquad j = 1, \ldots, N_n \tag{D.1}$$

$$h_j^l = \tanh a_j^l \qquad l = 1, \ldots, N_h, j = 1, \ldots, N_n \tag{D.2}$$

$$a_j^l = \sum_{i=1}^{N_n} w_{ji}^l h_i^{l-1} + b_j^l \qquad l = 2, \ldots, N_h, j = 1, \ldots, N_n \tag{D.3}$$

$$u_k = \sum_{i=1}^{N_n} W_{ki} h_i^{N_h} + B_k \qquad k = 1, \ldots, N_o \tag{D.4}$$

The model uses the following parameters:

- N_x is the number of inputs
- N_o is the number of outputs
- N_h is the number of intermediate layers

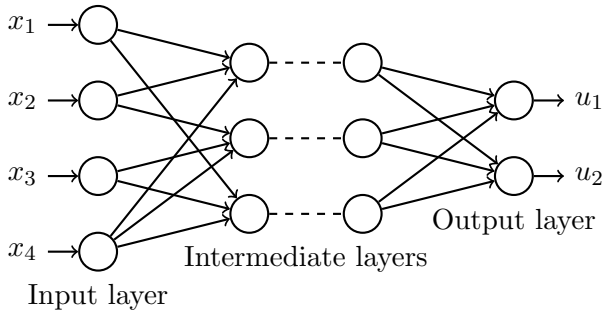


Figure D.1: Layered artificial neural network.

- N_n is the number of nodes in each intermediate layer
- $w_{ji^l}, b_j^l, W_{ki}, B_k$ are parameters specific to the ANN

Dua (2010) demonstrates how to tune the ANN parameters by minimizing the total prediction error as well as the ANN complexity. This tuning step is performed by solving a *training* MINLP. We denote CDUANN(x, u) the set of constraints defining the relation between the ANN inputs x and outputs u. In particular, the model inputs include crude properties q_C^{ip} and CDU cut points τ^k ($k \in \{\text{naphta}, \text{kerosene}, \text{diesel}\}$), and the model outputs include cut yields α^{ijk} and crude cut properties q_1^{ijkp}. All crude property inputs are fixed while CDU cut points are variable. The cut point between diesel and residue cuts can take three discrete values defining the three CDU operating modes. All the outputs are variable. The full refinery planning model is expressed as follows.

$$\max \quad \sum_{l \in L} p^l x_S^l \qquad \text{(sales revenue maximization)}$$

$$\text{s.t.} \quad 0 \leq \sum_{j \in J} x_F^{ij} \leq C^i \qquad i \in I \qquad \text{(crude availability)}$$

$$\underline{\tau^k} \leq \tau^k \leq \overline{\tau^k} \qquad k \in K \qquad \text{(CDU cut point limits)}$$

$$\text{CDUANN}\left(\left\{q_C^{ip}, \tau^k\right\}, \left\{\alpha^{ijk}, q_1^{ijkp}\right\}\right) \qquad \text{(CDU model)}$$

$$\underline{FR} \cdot H \leq \sum_{i \in I} \sum_{j \in J} x_F^{ij} \leq \overline{FR} \cdot H \qquad \text{(CDU flowrate limitations)}$$

$$x_1^{ijk} = \alpha^{ijk} \cdot x_F^{ij} \qquad (i,j,k) \in I \times J \times K \qquad \text{(CDU yield calculation)}$$

$$\sum_{i \in I} x_1^{ijk} = \sum_{l \in L} x_2^{jkl} \qquad (j,k) \in J \times K \qquad \text{(pool mass balance)}$$

$$\sum_{i \in I} q_1^{ijkp} x_1^{ijk} = q_2^{jkp} \sum_{l \in L} x_2^{jkl} \qquad (j,k,p) \in J \times K \times P \qquad \text{(pool quality balance)}$$

<u>nonlinear</u>

$$\sum_{j \in J} \sum_{k \in K} x_2^{jkl} = x_S^l \qquad l \in L \qquad \text{(product mass balance)}$$

$$x_S^l \leq D^l \qquad l \in L \qquad \text{(maximum product demand)}$$

$$\sum_{j \in J} \sum_{k \in K} q_2^{jkp} x_2^{jkl} \leq Z^{lp} x_S^l \qquad (l,p) \in L \times P \qquad \text{(product quality requirement)}$$

<u>nonlinear</u>

$$x_F^{ij}, x_1^{ijk}, x_2^{jkl}, x_S^l, \alpha^{ijk} \geq 0$$

$$q_1^{ijkp}, q_2^{jkp}, \tau^k \in \mathbb{R}$$