A Symmetry-Breaking Tool for the Optimization of Crude-Oil Operations Scheduling

Sylvain Mouret, Ignacio E. Grossmann and Pierre Pestiaux

Entreprise-Wide Optimization

Introduction

Goal

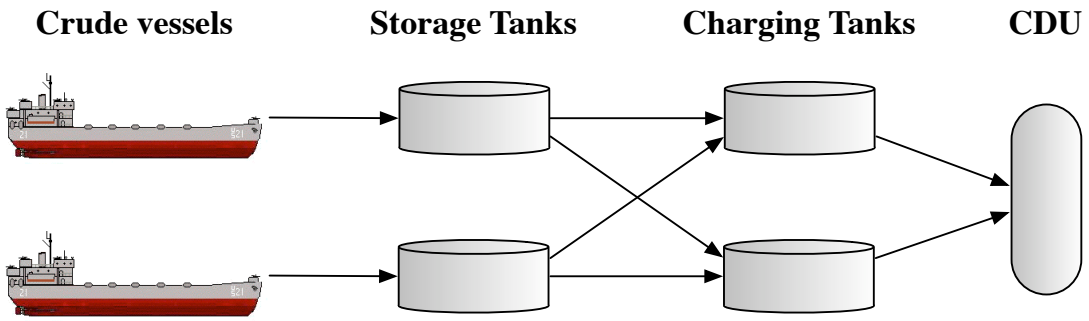
- Optimize the schedule of operations for the crude-oil unloading and blending problem using a continuous-time mathematical formulation
- Reduce the computation expense by breaking the symmetries in the MINLP model

Motivations

- Impacts the downstream refinery processes
- Multimillion dollar annual benefits application
- Kelly and Mann (2003)

Crude-oil operations scheduling problem

- Scheduling horizon $[0, H]$
- 4 types of resources:
  - Crude-oil marine vessels
  - Storage tanks
  - Charging tanks
  - Crude Distillation Units (CDUs)
- 3 types of operations:
  - Unloading: Vessel unloading to storage tanks
  - Transfer: Transfer from storage tanks to charging tanks
  - Distillation: Distillation of charging tanks



Problem definition

Given

- Refinery configuration
- Logistics constraints
- Initial tank inventory and composition
- Vessel arrival time, inventory level and composition
- Distillation specifications and demands (planning decisions)

Determine

- Required operations
- Timing decisions
- Transfer volumes

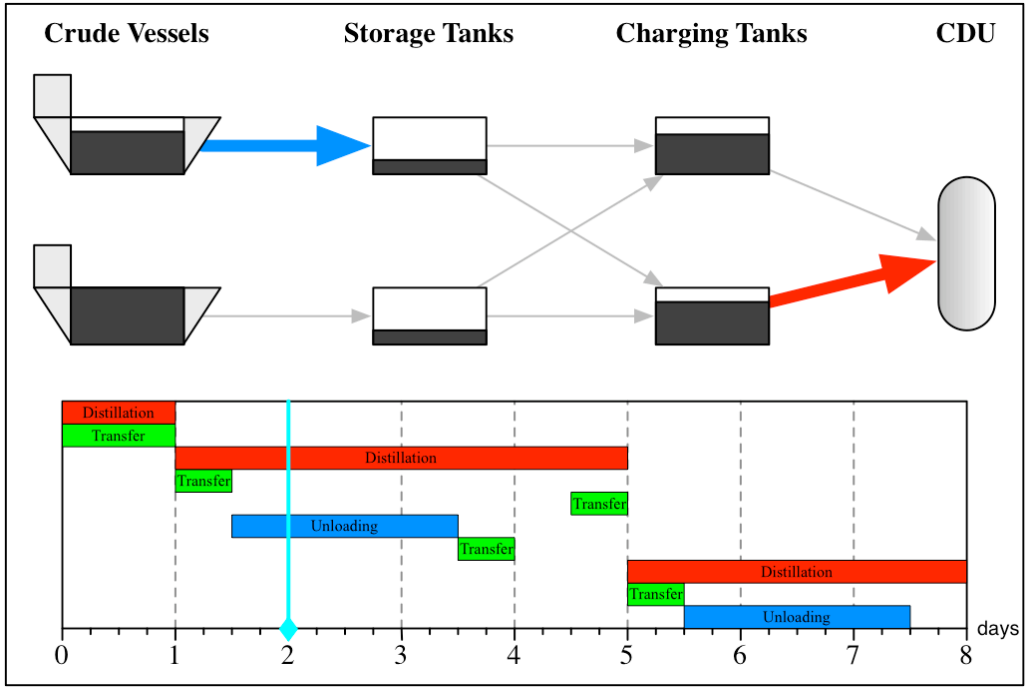
Minimize

- Costs of distilled crude-oil mixtures

Example of crude-oil operations schedule

Logistics constraints

- Only one docking station available for vessel unloadings
- A tank is either being filled, discharging, or idle
- A tank can charge only one CDU
- A CDU can be charged by only one tank
- Continuous distillation



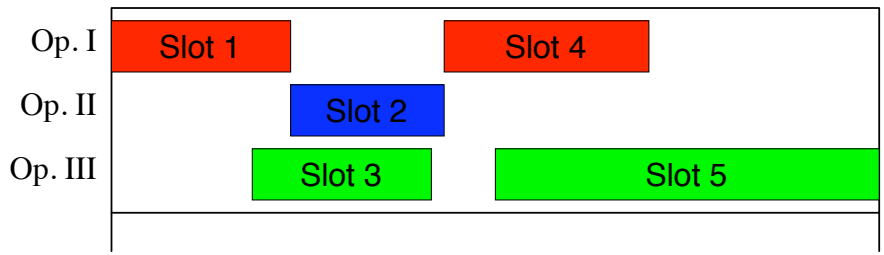
Refinery operations

Gantt chart

MINLP model: Basic Idea

Basic steps

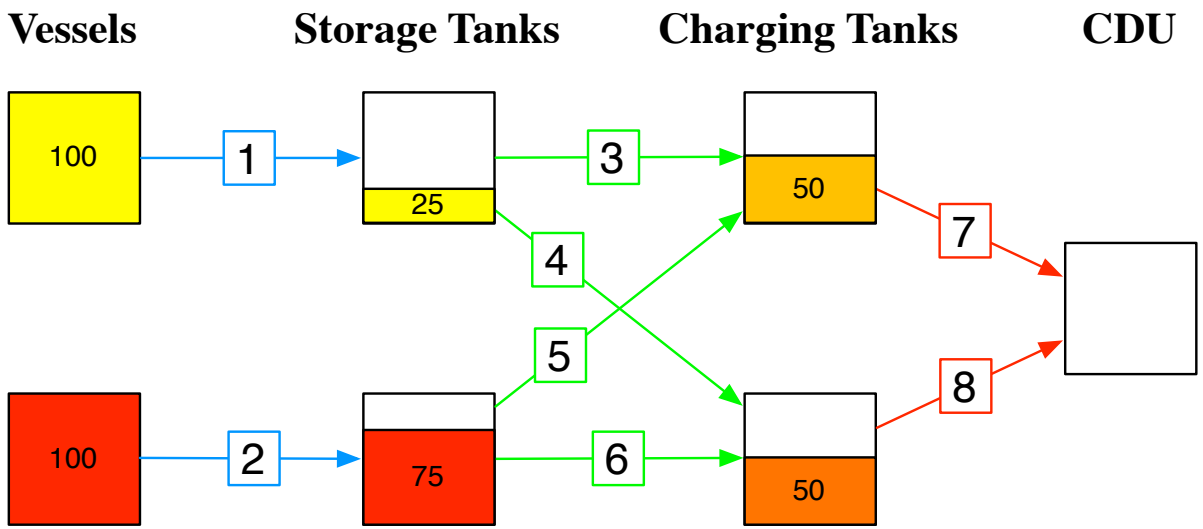
1. **Define** the set of all transfer operations
2. **Postulate** the number of time-slots
3. **Define** and order on the set of time-slots
4. **Assign** exactly one operation to each time-slot and **determine** the timing and volume decisions

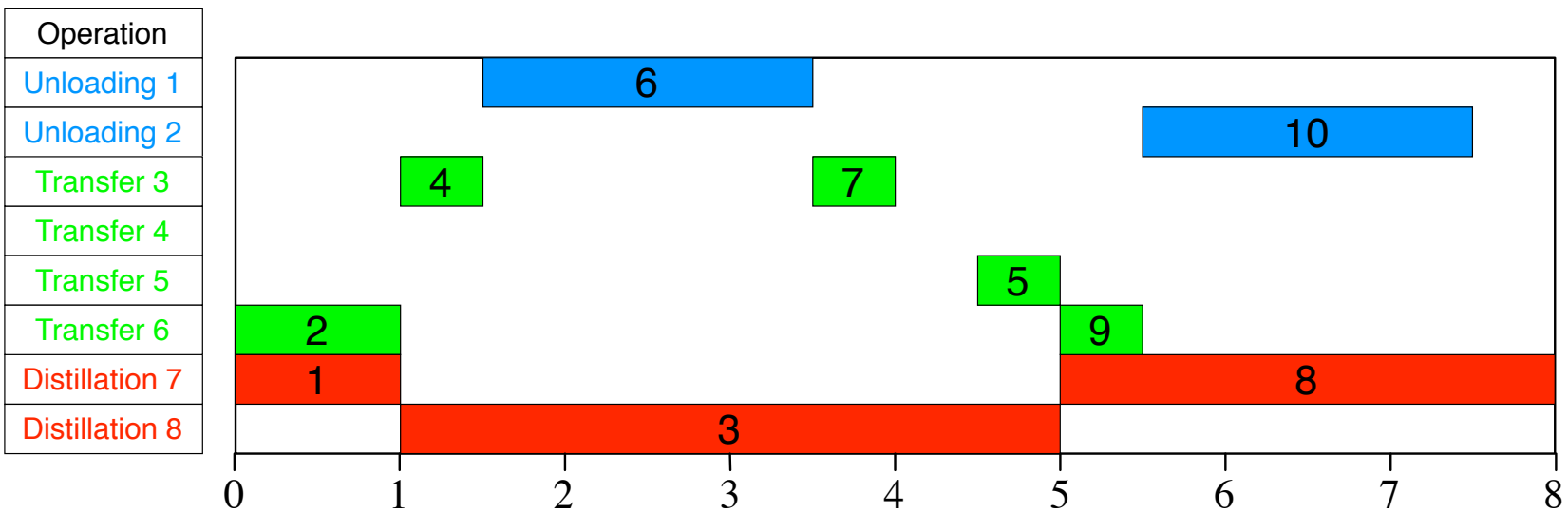


MINLP model

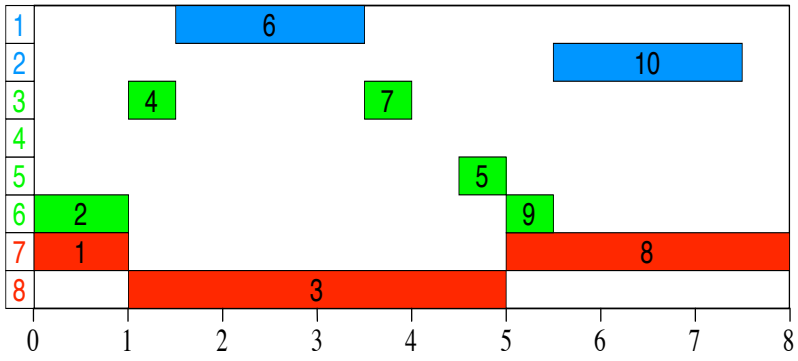
- Binary variables: assignment variables
- Continuous variables: time, volume and level variables

An example of operation assignment

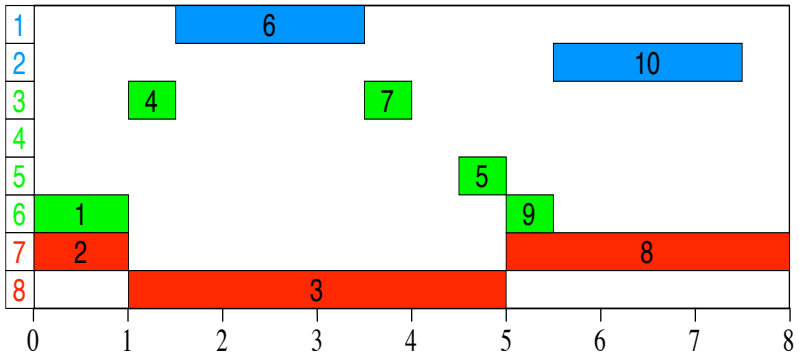




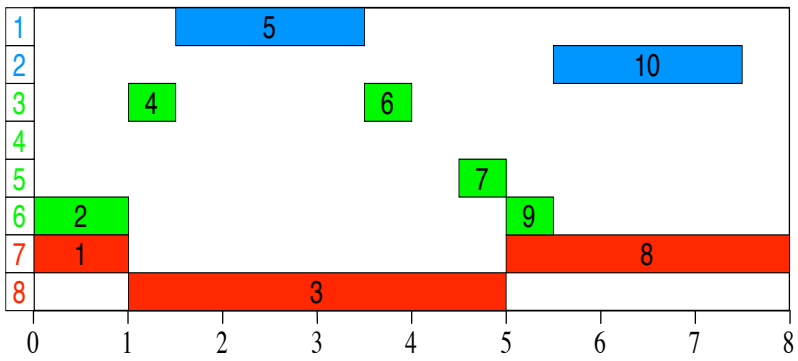
Symmetrical sequences of operations



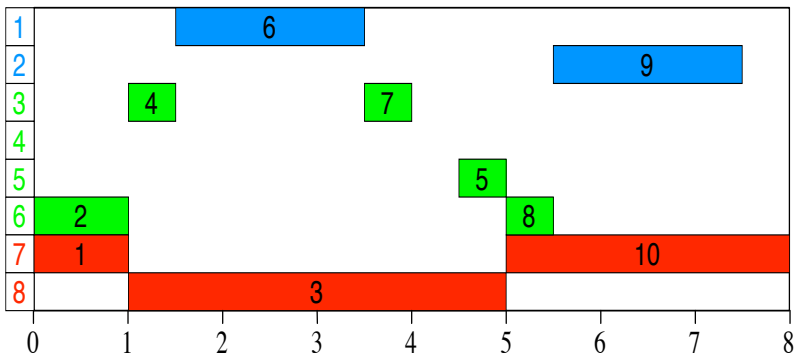
7683513762



6783513762



768**3135**762



7683513**627**

Breaking symmetries

Main Idea

- Define an **assignment rule** to restrict the number of discrete solutions for the same schedule
- A regular language and its corresponing deterministic finite automaton (DFA) are used to express the rule
- The regular language membership constraint on the sequence of operations is derived from the DFA as a system of linear equations (Côté et al., 2007) which can be added to the original model

Definitions

- Alphabet Σ

$$\Sigma = \{a, b, c\}$$

Definitions

- Alphabet Σ
- String w

$\Sigma = \{a, b, c\}$

$w = abbac$, $w = \varepsilon$ (empty string)

Definitions

- Alphabet Σ
- String w
- Set of all strings Σ^*

$\Sigma = \{a, b, c\}$

$w = abbac$, $w = \varepsilon$ (empty string)

Definitions

- Alphabet Σ — $\Sigma = \{a, b, c\}$
- String w — $w = abbac$, $w = \varepsilon$ (empty string)
- Set of all strings Σ^*
- Language $L \subset \Sigma^*$ — $L_1 = \{\varepsilon, a, b\}, L_2 = \{abba\}$

Definitions

- Alphabet Σ
- String w
- Set of all strings Σ^*
- Language $L \subset \Sigma^*$
- Set union $L_1 + L_2$

$\Sigma = \{a, b, c\}$

$w = abbac,\ w = \varepsilon$ (empty string)

$L_1 = \{\varepsilon, a, b\}, L_2 = \{abba\}$

$L_1 + L_2 = \{w / w \in L_1 \vee w \in L_2\}$

$L_1 + L_2 = \{\varepsilon, a, b, abba\}$

Definitions

- Alphabet Σ — $\Sigma = \{a, b, c\}$
- String w — $w = abbac$, $w = \varepsilon$ (empty string)
- Set of all strings Σ^*
- Language $L \subset \Sigma^*$ — $L_1 = \{\varepsilon, a, b\}, L_2 = \{abba\}$
- Set union $L_1 + L_2$ — $L_1 + L_2 = \{w / w \in L_1 \vee w \in L_2\}$; $L_1 + L_2 = \{\varepsilon, a, b, abba\}$
- Concatenation $L_1 L_2$ — $L_1 L_2 = \{w / \exists w_1 \in L_1, w_2 \in L_2, w = w_1 w_2\}$; $L_1 L_2 = \{abba, aabba, babba\}$

Definitions

- Alphabet Σ
- String w
- Set of all strings Σ^*
- Language $L \subset \Sigma^*$
- Set union $L_1 + L_2$
- Concatenation $L_1 L_2$
- Kleene star L^*

$\Sigma = \{a, b, c\}$

$w = abbac,\ w = \varepsilon$ (empty string)

$L_1 = \{\varepsilon, a, b\}, L_2 = \{abba\}$

$L_1 + L_2 = \{w / w \in L_1 \vee w \in L_2\}$

$L_1 + L_2 = \{\varepsilon, a, b, abba\}$

$L_1 L_2 = \{w / \exists w_1 \in L_1, w_2 \in L_2, w = w_1 w_2\}$

$L_1 L_2 = \{abba, aabba, babba\}$

$L^* = \{\varepsilon\} \cup L \cup LL \cup LLL \cup \ldots$

$L_2^* = \{\varepsilon, abba, abbaabba, \ldots\}$

Class of Regular Languages $\mathcal{R}eg$

Definition of the class $\mathcal{R}eg$

The class of Regular Languages, noted $\mathcal{R}eg$, is defined as follows:

$$\emptyset \in \mathcal{R}eg$$
$$\{\varepsilon\} \in \mathcal{R}eg$$
$$\{a\} \in \mathcal{R}eg \qquad \forall a \in \Sigma$$
$$L_1 + L_2 \in \mathcal{R}eg \qquad \forall L_1, L_2 \in \mathcal{R}eg$$
$$L_1 L_2 \in \mathcal{R}eg \qquad \forall L_1, L_2 \in \mathcal{R}eg$$
$$L^* \in \mathcal{R}eg \qquad \forall L \in \mathcal{R}eg$$

- A Regular Language is built from the alphabet Σ and the symbols "ϵ", "(", ")", "$+$", and "$*$".

Some regular languages

- $a(\varepsilon + b) = \{a, ab\} \in \mathcal{R}eg$
  $a(\varepsilon + b)$ contains the two strings a and ab

- $a^*b^* = \{\varepsilon, a, b, aa, ab, bb, \ldots\} \in \mathcal{R}eg$
  a^*b^* contains the strings built from a succesion of a's then b's

- $\{a^n b^n / n \in \mathbb{N}\} \notin \mathcal{R}eg$

- $o^*(aa^*oo^* + bb^*ooo^*)^* = \{\varepsilon, o, ao, oao, aoo, boo, \ldots\} \in \mathcal{R}eg$

Definition

Definition of a Deterministic Finite Automaton (DFA)

A DFA is defined by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:

- Q is a finite set of states
- Σ is an alphabet
- $\delta : Q \times \Sigma \to Q$ is a partial transition function
- $q_0 \in Q$ is the initial state
- $F \subset Q$ is the set of final states

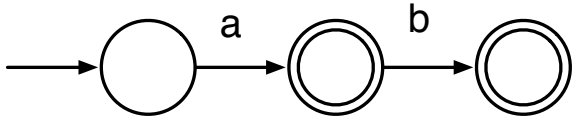
Given an input string, the automaton starts in the initial state q_0 and process the string one symbol at a time, applying the transition function δ at each step to update the current state.
The string is accepted if and only if the last state reached belongs to F.

Some DFAs with associated regular languages

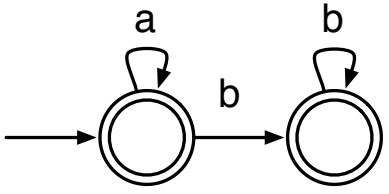
Theorem

A language is regular iff it is recognized by a DFA.

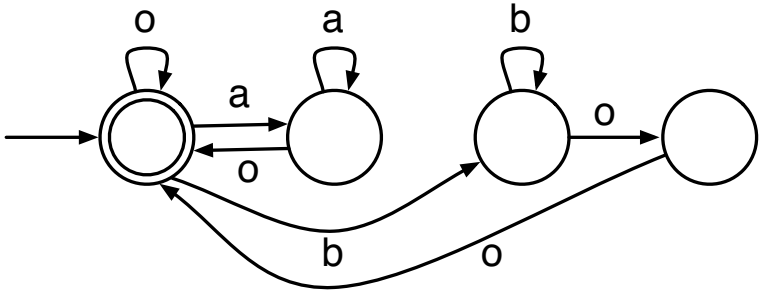
$a(\varepsilon + b)$



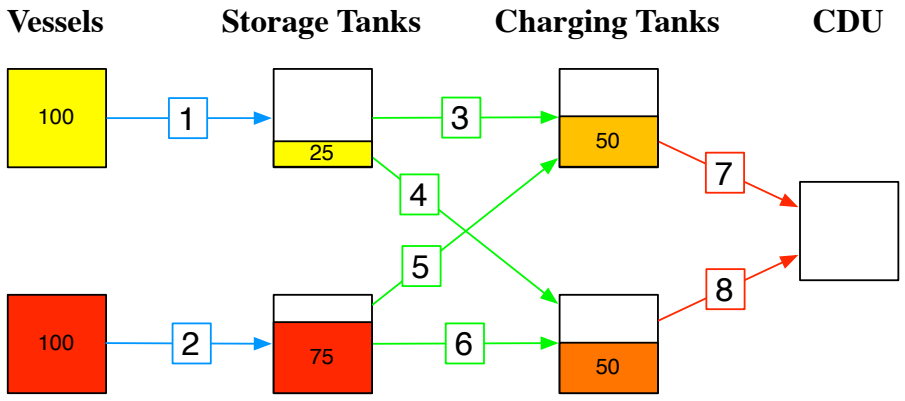
$a^* b^*$



$o^*(aa^*oo^* + bb^*ooo^*)^*$

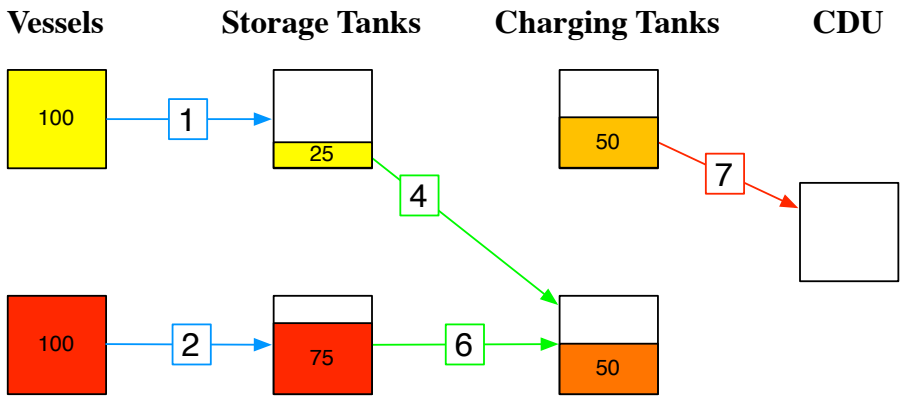


Rule derivation



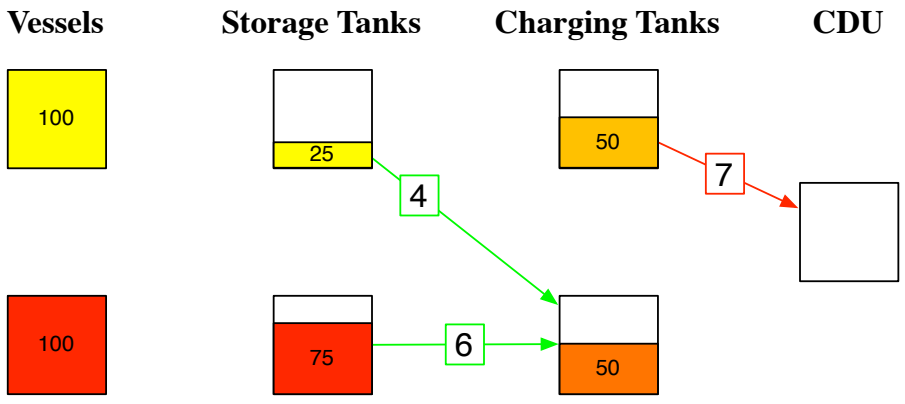
- The sequence of operations can be decomposed into sub-sequences for **distillation states** 7 and 8

Rule derivation



- The sequence of operations can be decomposed into sub-sequences for **distillation states** 7 and 8
- Let L_7 be the regular language for distillation state 7 (L_8 for state 8)

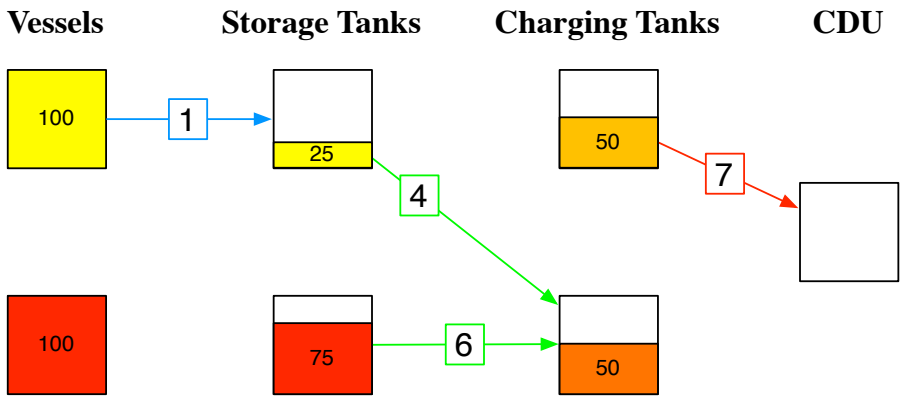
Rule derivation



- The sequence of operations can be decomposed into sub-sequences for **distillation states** 7 and 8
- Let L_7 be the regular language for distillation state 7 (L_8 for state 8)
- If no unloading operation is performed:

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

Rule derivation



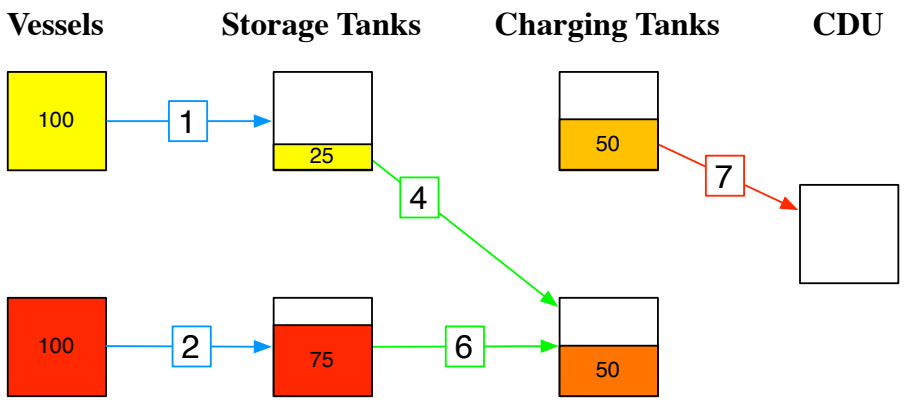
- The sequence of operations can be decomposed into sub-sequences for **distillation states** 7 and 8
- Let L_7 be the regular language for distillation state 7 (L_8 for state 8)
- If no unloading operation is performed:

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

- If only unloading operation 1 may be performed:

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

Rule derivation



- The sequence of operations can be decomposed into sub-sequences for **distillation states** 7 and 8
- Let L_7 be the regular language for distillation state 7 (L_8 for state 8)
- If no unloading operation is performed:

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

- If only unloading operation 1 may be performed:

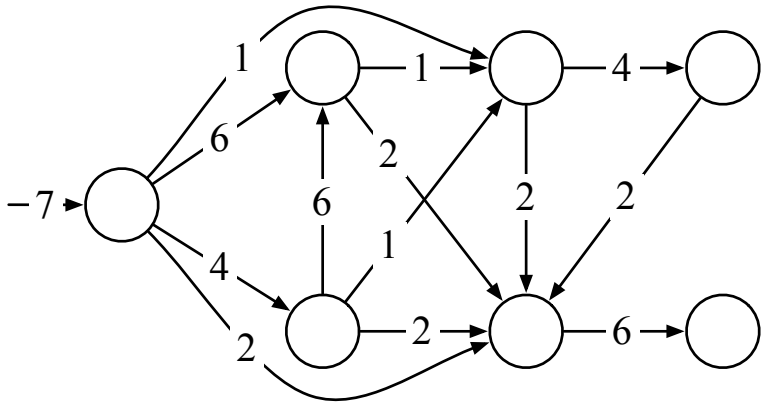
$$L_7 = 7(\varepsilon + 4)(\varepsilon + 6)(\varepsilon + 1 + 14)$$

- If both unloading operations 1 and 2 may be performed:

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

Rule derivation

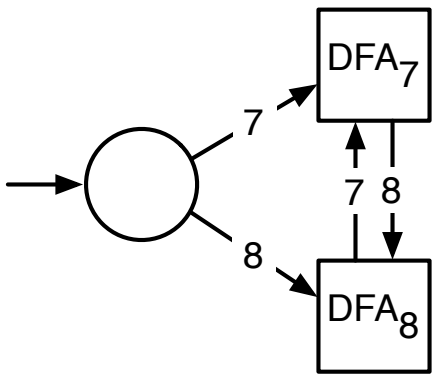
- DFA recognizing the regular language L_7



- Regular language L

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

- DFA recognizing the regular language L



Regular language membership constraint

The regular language membership constraint, initially developped by Pesant (2003) for CP, has been adapted by Côté et al. (2007) for MILP. It is formulated as a network flow problem relying on a layered directed graph.

$$Z_{iv} = \sum_{q} S_{ivq}$$

$$\sum_{v,q=\delta(q_0,v)} S_{1vq_0} = 1$$

$$\sum_{v,q',q=\delta(q',v)} S_{(i-1)vq'} = \sum_{v,q'=\delta(q,v)} S_{ivq} \qquad \forall i, q$$

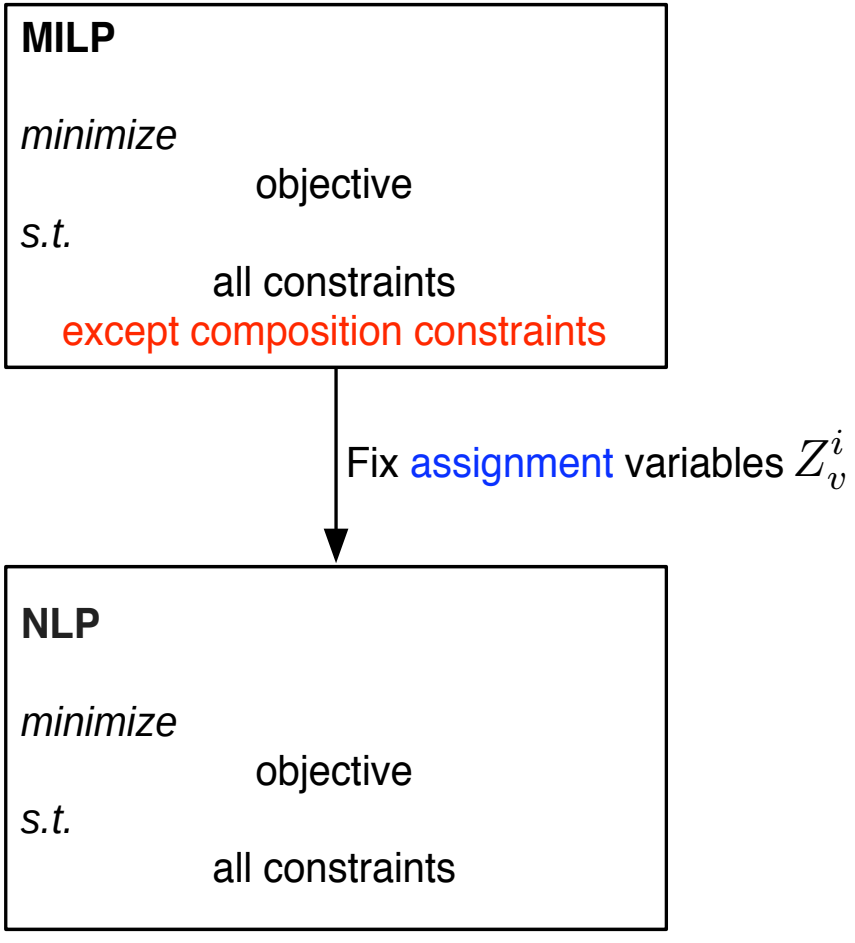
$$\sum_{v,q',q=\delta(q',v)} S_{nvq'} = SF_q \qquad \forall q \in F$$

$$\sum_{q} SF_q = 1$$

MILP-NLP decomposition

Decomposition steps

1. **Solve** the MILP relaxation
2. **Check** feasibility: the solution may not satisfy the nonlinear composition constraints
3. If infeasible, **Fix** assignment variables
4. **Solve** the resulting NLP (with nonlinear composition constraints)



Computational Results

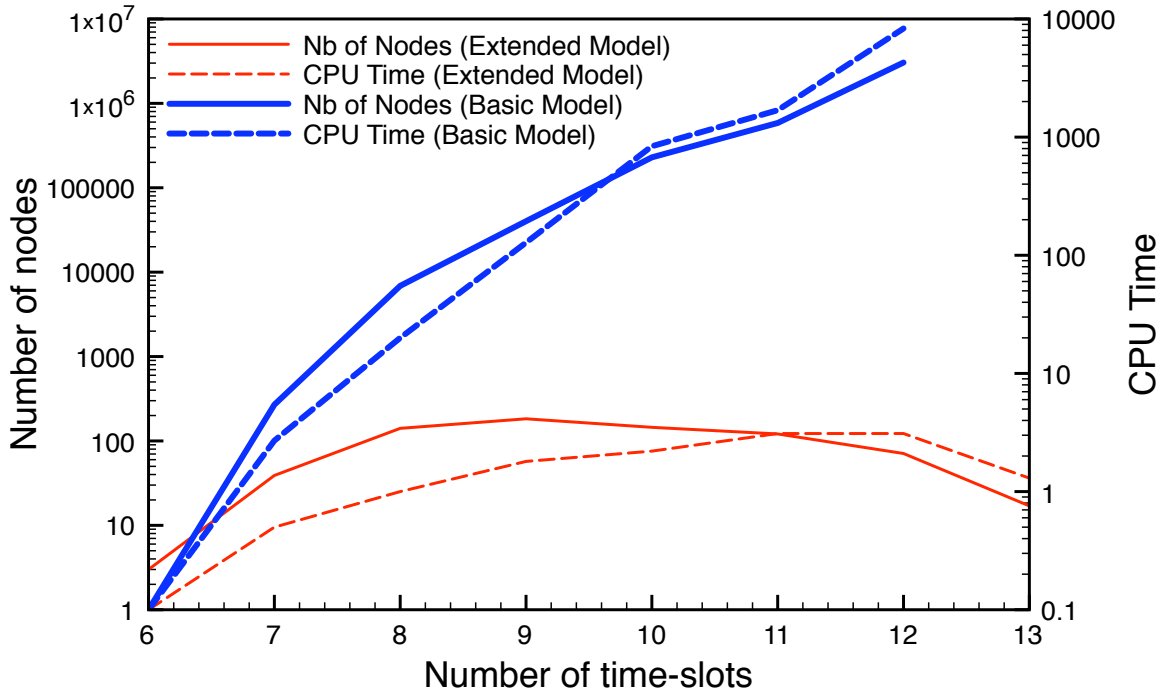
- Approach tested on the 4 problems from Lee et al. (1996) using Xpress (MILP) and CONOPT (NLP)
- Problems solved for different number of time-slots

Pb	Vess./Stor./Charg./CDUs	Operations	Slots	MILP	NLP	Gap	CPU
1	2 / 2 / 2 / 1	8	13	120.25	120.25	0%	1s
2	3 / 3 / 3 / 2	14	21	198.83	198.83	0%	26s
3	3 / 3 / 3 / 2	14	21	59.60	62.50	4.9%	63s
4	3 / 6 / 4 / 3	19	26	107.47	107.47	0%	112s

- Basic MILP model without symmetry-breaking constraints -12 slots
  - Size: 1,259 variables (96 binary), 3,303 constraints
  - CPU time: 8,307s
  - Number of nodes: 3,045,400
- Extended MILP model with symmetry-breaking constraints -12 slots
  - Size: 2,714 variables (1,455 binary), 3,581 constraints
  - CPU time: 3s
  - Number of nodes: 71

Comparison Basic/Extended models

- Problems 1 tested with 6 to 13 time-slots
- Basic model
  - The search space and CPU time grow exponentialy with the size of the input
- Extended model
  - The search space and CPU time reach a maximum before decreasing
  - The number of nodes reaches a maximum for 9 time-slots
  - The CPU time reaches a maximum for 11 time-slots



Conclusion and future work

Conclusion

- New MINLP formulation for the crude-oil operations problem
- Handles logistics constraints and minimization crude-oil costs
- Includes DFA-based symmetry-breaking constraints relying on the structure of the scheduling problem
- MILP-NLP decomposition algorithm compares well to MINLP solvers

Conclusion and future work

Conclusion

- New MINLP formulation for the crude-oil operations problem
- Handles logistics constraints and minimization crude-oil costs
- Includes DFA-based symmetry-breaking constraints relying on the structure of the scheduling problem
- MILP-NLP decomposition algorithm compares well to MINLP solvers

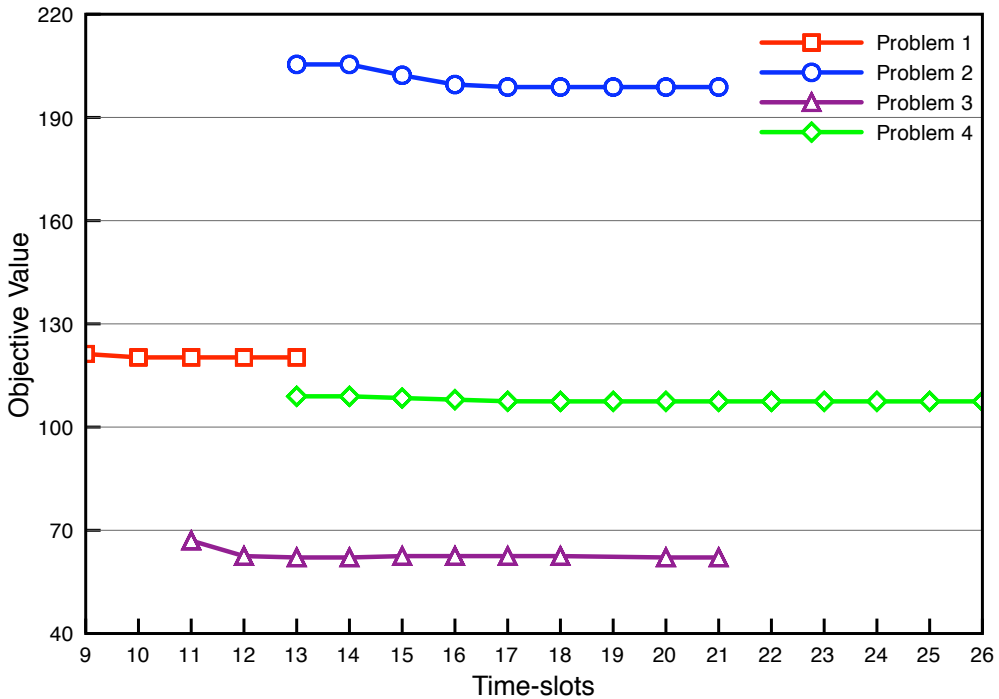
Future work

- Hybrid optimization: Contraint Programming as a symmetry-breaking branching tool
- Enhance the MILP-NLP decomposition (LP/NLP based branch and bound, Quesada and Grosmann, 1992)
- Take into account stochastic parameters (vessels arrival time)
- Practical case-study

Computational Results

- Approach tested on the 4 problems from Lee et al. (1996) using Xpress (MILP) and CONOPT (NLP)
- Problems solved for different number of time-slots

Pb	Vess./Stor./Charg./CDUs	Operations	Slots	MILP	NLP	Gap	CPU
1	2 / 2 / 2 / 1	8	13	120.25	120.25	0%	1s
2	3 / 3 / 3 / 2	14	21	198.83	198.83	0%	26s
3	3 / 3 / 3 / 2	14	21	59.60	62.50	4.9%	63s
4	3 / 6 / 4 / 3	19	26	107.47	107.47	0%	112s



Comparison with MINLP solvers

- Problem 1 and 2 tested with 13 and 21 time-slots
- Algorithms used:
  - MILP-NLP decomposition: Xpress (MILP), CONOPT (NLP)
  - MINLP solvers: DICOPT, SBB, AlphaECP, BARON

	Problem 1		Problem 2	
Algorithm	Solution	CPU time	Solution	CPU time
MINLP-NLP	120.25	1s	198.83	26s
DICOPT	120.25	37s	198.83	1310s
AlphaECP	120.25	111s	198.83	1092s
SBB	120.25	110s	–	+3600s
BARON	120.25	433s	–	+3600s

$\Rightarrow$ Order of magnitude reduction for CPU time.