NEW FORMULATIONS AND A COLUMN GENERATION-BASED HEURISTIC FOR A PROBLEM OF PETROLEUM TRANSPORTATION

Luiz Aizemberg, Hugo Harry Kramer, Artur Alves Pessoa, Eduardo Uchoa
Departamento de Engenharia de Produção - Universidade Federal Fluminense
Rua Passo da Pátria, 156, Bloco E, 4º andar, São Domingos, 24210-240, Niterói, RJ
{luizaizemberg, hugoharry}@gmail.com, {artur, uchoa}@producao.uff.br

ABSTRACT

In this paper, we study tactical models for a crude oil transportation problem. The problem considers inventory capacities and discrete lot sizes to be transported by tankers, aiming to supply given demands over a finite time horizon. We show a new formulation that achieves a better performance than the literature. The results are compared over 3 classes of instances. A column generation based heuristic is proposed to find good feasible solutions for the hardest class of instances with less computational burden than the heuristics of the commercial solver used.
KEY WORDS. Mathematical Programming, Logistics, Petroleum.

Main Areas: Optimization, Supply Chain Management.

1 Introduction

In this paper, we study a tactical optimization model for crude oil distribution by tankers. The problem consists of scheduling the shipments through routes linking platforms (offshore production sites) and terminals (onshore consumer sites). The routes are assumed to be round trips, where a tanker departs loaded from a platform, offloads at a terminal and travels back to the platform of origin unloaded. The objective is to ship a single product from the platforms to supply the terminals with minimum transportation cost for a finite planning horizon. For each site, the inventory levels must lie between a lower and an upper bound to avoid the lack or excess of product. Also, for each site, the demand or the production amounts are given for the whole planning horizon. The product is shipped only by oil tankers and we assume that the global capacity of the fleet is unlimited. We also consider that when an oil tanker leaves a platform, it is loaded with his full capacity, and there are economies of scale, i. e., larger tankers have smaller unitary costs. For each transportation, one has to decide the route, the lot size and the delivery day.

The problem as it is defined here has already been addressed in Rocha *et al*. (2011) and Aizemberg *et al*. (2012). Rocha *et al*. (2011) present some formulations for the problem, and the formulation with the best performance is referred to as *Knapsack Cascading*. Aizemberg *et al*. (2012) also study some formulations for the problem, and the best formulation proposed, which outperforms the results of previous works, is the *Knapsack Cascading* formulation tightened by rounding of right hand sides, and with new variables representing accumulated shipping in routes. In the present work, a new formulation that outperforms those found in literature for almost all instances is devised. The instances used as benchmark are those proposed by Rocha *et al*. (2011) (medium and hard instances) and Aizemberg *et al*. (2012) (harder instances).

A column generation based heuristic approach is presented and tested only over the harder class of instances. We note that, for this class of instances, the commercial solver used was not capable of finding good feasible solutions at the beginning of the branch-and-bound algorithm tree. Finding a good feasible solution at the very beginning of the tree is crucial to reduce the search by cutting off nodes of such tree, increasing the probability of proving the optimality in a reasonable computational time.

1.1 Literature review

Several models are found in literature to deal with problems involving the transportation of oil derivatives, but considering a wide range of different characteristics. To optimize such models, Mixed Integer Programming (MIP) techniques are the most used. In Velez and Maravelias (2013), the authors deal with the scheduling of chemical production which typically has many symmetric solutions. Three MIP formulations are proposed and additional constraints are created to define the number of batches of each task as an integer variable. Branching on this new integer variable leads to the elimination of many symmetric solutions. In Magatão *et al*. (2004) and Boschetto *et al*. (2010), MIP approaches are applied to schedule the activities in a real pipeline network. The former uses illustrative instances with four products, while the latter uses large instances, where more than 14 oil derivatives and ethanol are transported. Finally, Banaszewski *et al*. (2010) and Aizemberg *et al*. (2011) deal with the medium-term planning of oil derivatives transportation in the multimodal supply chain network of a Brazilian oil/energy company. The former applied an auction based multiagent algorithm, while the latter applied a MIP based local search approach over a mathematical formulation.

Column generation-based heuristics (CGH) are found in literature to deal with many different problems: multi-item scheduling (Bahl, 1983), vehicle routing (Mourgaya and Vanderbeck, 2007), cutting stock (Furini *et al*., 2012), sensor placement (Yavuz *et al*., 2010), generalized assignment (Moccia *et al*., 2009) and many others. In general, such approaches share similar main characteristics: first, the linear programming relaxation is solved by column generation. Then, a feasible solution is constructed for the original problem using the obtained columns. For example,

in Choi and Tcha (2007) the authors apply a column generation heuristic for the heterogeneous fleet vehicle routing problem which consists on three steps: (i) solve the continuous relaxation; (ii) set the variables of the final restricted master as binaries; and (iii) solve the resulting MIP problem. In Joncour *et al*. (2010), the authors review generic classes of column generation-based heuristics.

1.2 Paper organization

The remainder of the paper is organized as follows. Section 2 presents the problem definitions and the mathematical formulations. Section 3 shows the column generation based heuristic. Section 4 discusses the obtained results. At last, Section 5 contains the conclusions of this work.

2 Mathematical formulations

The optimization problem defined in Rocha *et al*. (2011) is defined as follows. We are given a set P of platforms, a set T of terminals, a set C of classes of oil tankers and the length D of the planning horizon in days. Let $T(p) \subseteq T$ be the subset of terminals that are allowed to receive a tanker from a platform $p \in P$, and let $P(t) \subseteq P$ be the subset of platforms allowed to send a tanker to a terminal $t \in T$. A single product should be shipped from platforms to supply the demands of the terminals. The inventory levels of platforms and terminals are bounded by lower and upper bounds, i.e., the lack or surplus of the product are not allowed at any point of the network during the whole planning horizon. The production amount $P_{p,d}$ of a platform $p \in P$ at day d, the maximum inventory capacity CAP_p and initial inventory $e_{p,0}$ are given, as well as the demand $C_{t,d}$ of a terminal $t \in T$ at day d, the maximum inventory capacity CAP_t and initial inventory $e_{t,0}$. Minimum inventory capacities are assumed to be zero. A class $c \in C$ contains oil tankers with same shipping capacity V_c and transportation cost per day F_c, and it is assumed that only oil tankers with fulfilled capacities are used in the transportation. The transportation time between a platform $p \in P$ and a terminal $t \in T(p)$ is given by $D_{p,t}$. The objective is to minimize the total transportation costs.

Some additional definitions are listed on the following. Let L_p be the greatest common divisor (GCD) of all shipping capacities of all classes of tankers that are allowed to depart from a platform $p \in P$, and let L_t be the GCD of all shipping capacities of all classes of tankers that are allowed to supply terminal $t \in T$. In addition, let $AC_{p,d}$ be the accumulated production of a platform $p \in P$ from day 1 to a day d plus its initial inventory $e_{p,0}$, and let $AC_{t,d}$ be the accumulated demand of terminal $t \in T$ from day 1 to a day d, minus its initial inventory $e_{t,0}$.

2.1 Rounded Knapsack Cascading formulation

The Rounded Knapsack Cascading ($\mathcal{RKC}$) formulation is a stronger version of the $\mathcal{KC}$ formulation of Rocha *et al*. (2011), tightened by rounding the right hand sides of the constraints. Let the binary variables $z_{p,t}^{c,d}$ indicate if an oil tanker of class $c \in C(p)$ is sent from a platform $p \in P$ to a terminal $t \in T(p)$ at day d, $d \in \{1, ..., D\}$. We assume that $z_{p,t}^{c,d} = 0$ whenever $d < 1$. Thus, the $\mathcal{RKC}$ formulation is given by:

$$(\mathcal{RKC}) \min \sum_{p \in P} \sum_{t \in T(p)} \sum_{c \in C(p)} \sum_{d=1}^{D} 2F_c D_{p,t} z_{p,t}^{c,d} \tag{1}$$

subject to

$$\sum_{t \in T(p)} \sum_{\tau=1}^{d} \sum_{c \in C(p)} V_c z_{p,t}^{c,\tau} \leq \left\lfloor \frac{AC_{p,d}}{L_p} \right\rfloor L_p \quad \forall p \in P, d \in \{1, ..., D\} \tag{2}$$

$$\sum_{t\in T(p)}\sum_{\tau=1}^{d}\sum_{c\in C(p)} V_c z_{p,t}^{c,\tau} \geq \left\lceil \frac{AC_{p,d} - CAP_p}{L_p} \right\rceil L_p \quad \forall p \in P, d \in \{1,...,D\} \tag{3}$$

$$\sum_{p\in P(t)}\sum_{\tau=1}^{d}\sum_{c\in C(p)} V_c z_{p,t}^{c,\tau-D_{p,t}} \geq \left\lceil \frac{AC_{t,d}}{L_t} \right\rceil L_t \quad \forall t \in T, d \in \{1,...,D\} \tag{4}$$

$$\sum_{p\in P(t)}\sum_{\tau=1}^{d}\sum_{c\in C(p)} V_c z_{p,t}^{c,\tau-D_{p,t}} \leq \left\lfloor \frac{CAP_t + AC_{t,d}}{L_t} \right\rfloor L_t \quad \forall t \in T, d \in \{1,...,D\} \tag{5}$$

$$z_{p,t}^{c,d} \in \{0,1\} \quad \forall p \in P, t \in T(p), c \in C(p), d \in \{1,...,D\}. \tag{6}$$

The objective function (1) minimizes the total transportation cost, including the trip back from the terminal to the platform after each transportation. Constraints (2) avoid the inventory levels at each platform and each day to be less than the minimum capacity. Constraints (3) ensure that the inventory levels at each platform never exceed their maximum inventory capacity at each day. Constraints (4) avoid the inventory levels at each terminal and each day to be less than the minimum capacity and constraints (5) ensure that the inventory levels at each terminal never exceed their maximum inventory capacity at each day. Finally, the z variables are defined by (6) as binaries.

2.2 Rounded Cascading Accumulated in Routes formulation

This formulation is adapted from Aizemberg *et al.* (2012). It differs from the $\mathcal{RKC}$ formulation by the addition of integer variables $x_{p,t}^{c,d}$ representing the number of tankers of class $c \in C(p)$ leaving a platform $p \in P$ to a terminal $t \in T(p)$ from day 1 to day d, and assume that $x_{p,t}^{c,0} = 0$. The Rounded Cascading Accumulated in routes ($\mathcal{RCAR}$) formulation is given by:

$$(\mathcal{RCAR}) \min \sum_{p\in P}\sum_{t\in T(p)}\sum_{c\in C(p)} 2F_c D_{p,t} x_{p,t}^{c,D} \tag{7}$$

subject to

$$x_{p,t}^{c,d} = x_{p,t}^{c,d-1} + z_{p,t}^{c,d} \quad \forall p \in P, t \in T(p), c \in C(p), d \in \{1,...,D\} \tag{8}$$

$$\sum_{t\in T(p)}\sum_{c\in C(p)} V_c x_{p,t}^{c,d} \leq \left\lfloor \frac{AC_{p,d}}{L_p} \right\rfloor L_p \quad \forall p \in P, d \in \{1,...,D\} \tag{9}$$

$$\sum_{t\in T(p)}\sum_{c\in C(p)} V_c x_{p,t}^{c,d} \geq \left\lceil \frac{AC_{p,d} - CAP_p}{L_p} \right\rceil L_p \quad \forall p \in P, d \in \{1,...,D\} \tag{10}$$

$$\sum_{p\in P(t)}\sum_{c\in C(p)} V_c x_{p,t}^{c,d-D_{p,t}} \geq \left\lceil \frac{AC_{t,d}}{L_t} \right\rceil L_t \quad \forall t \in T, d \in \{1,...,D\} \tag{11}$$

$$\sum_{p\in P(t)}\sum_{c\in C(p)} V_c x_{p,t}^{c,d-D_{p,t}} \leq \left\lfloor \frac{CAP_t + AC_{t,d}}{L_t} \right\rfloor L_t \quad \forall t \in T, d \in \{1,...,D\} \tag{12}$$

$$x_{p,t}^{c,d} \in \mathbb{Z}_+, z_{p,t}^{c,d} \in [0,1] \quad \forall p \in P, t \in T(p), c \in C(p), d \in \{1,...,D\} \tag{13}$$

The objective function (7) is to minimize the total transportation cost. Constraints (8) define the x variables in terms of the z variables. Constraints (9) and (10) ensure that the inventory

level in each platform p at day d is not less than the minimum inventory capacity and does not exceed the maximum inventory capacity. Constraints (11) and (12) avoid the inventory level in each terminal t at day d to be less than the minimum capacity and to exceed the maximum inventory capacity. At last, constraints (13) define the x variables as non negative integers and the z variables as continuous variables in the range between zero and one. Note that constraint (8) ensures the integrality of the z variables when all x values are integer.

In this formulation, as pointed out by Aizemberg *et al*. (2012), by the addition of the x variables, the number of non-zero coefficients of the constraints matrix is much smaller than in $\mathcal{RKC}$ because each x variable replaces a sum of many z variables. Moreover, the x variables are more suitable for branching.

2.3 Rounded Cascading Accumulated in Sites formulation

This new formulation also differs from $\mathcal{RKC}$ formulation by the addition of new integer variables. These new variables are defined as follows: let $x_p^{c,d}$ be an integer variable representing the number of tankers of class $c \in C(p)$ leaving platform $p \in P$ from day 1 to day d, and let $x_t^{c,d}$ be an integer variable representing the number of tankers of class $c \in C(t)$ arriving at terminal $t \in T$ from day 1 to day d. Assume that $x_p^{c,0} = x_t^{c,0} = 0$. Given this, the Rounded Cascading Accumulated in Sites ($\mathcal{RCAS}$) formulation is given by:

$$(\mathcal{RCAS}) \min \sum_{p\in P} \sum_{t\in T(p)} \sum_{c\in C(p)} \sum_{d=1}^{D} 2F_c D_{p,t} z_{p,t}^{c,d} \tag{14}$$

subject to

$$x_p^{c,d} = x_p^{c,d-1} + \sum_{t\in T(p)} z_{p,t}^{c,d} \quad \forall p \in P, c \in C(p), d \in \{1,...,D\} \tag{15}$$

$$x_t^{c,d} = x_t^{c,d-1} + \sum_{p\in P(t)} z_{p,t}^{c,d-D_{p,t}} \quad \forall t \in T, c \in C(t), d \in \{1,...,D\} \tag{16}$$

$$\sum_{c\in C(p)} V_c x_p^{c,d} \leq \left\lfloor \frac{AC_{p,d}}{L_p} \right\rfloor L_p \quad \forall p \in P, d \in \{1,...,D\} \tag{17}$$

$$\sum_{c\in C(p)} V_c x_p^{c,d} \geq \left\lceil \frac{AC_{p,d} - CAP_p}{L_p} \right\rceil L_p \quad \forall p \in P, d \in \{1,...,D\} \tag{18}$$

$$\sum_{c\in C(t)} V_c x_t^{c,d-D_{p,t}} \geq \left\lceil \frac{AC_{t,d}}{L_t} \right\rceil L_t \quad \forall t \in T, d \in \{1,...,D\} \tag{19}$$

$$\sum_{c\in C(t)} V_c x_t^{c,d-D_{p,t}} \leq \left\lfloor \frac{CAP_t + AC_{t,d}}{L_t} \right\rfloor L_t \quad \forall t \in T, d \in \{1,...,D\} \tag{20}$$

$$x_p^{c,d} \in \mathbb{Z}_+ \quad \forall p \in P, c \in C(p), d \in \{1,...,D\} \tag{21}$$

$$x_t^{c,d} \in \mathbb{Z}_+ \quad \forall t \in T, c \in C(t), d \in \{1,...,D\} \tag{22}$$

$$z_{p,t}^{c,d} \in [0,1] \quad \forall p \in P, t \in T(p), c \in C(p), d \in \{1,...,D\}. \tag{23}$$

Note that, if all x variables are fixed, constraints (15) and (16) define an instance of the classical transportation problem (the remaining constraints contain only x variables). Thus, the integrality of z variables is ensured for any solution where all x variables are integer.

3 Column Generation-based Heuristic

This section presents the proposed heuristic based on Column Generation (CG). First we describe the Dantzig-Wolfe decomposition and the pricing subproblem proposed by Aizemberg *et al.* (2012). Next, we show a dynamic programming recursion to solve the pricing subproblem. Lastly, a pseudocode describes the heuristic procedure, combining the Dantzig-Wolfe decomposition and the $\mathcal{RCAS}$ formulation.

3.1 Dantzig-Wolfe decomposition

Here, we present a Dantzig and Wolfe (1960) decomposition for the problem using the ($\mathcal{RKC}$) formulation as starting point. The following notation is considered for the reformulation: let S_p be the set of all vectors of feasible solutions for a platform $p \in P$, let $V_{p,s}^{t,d}$ be the amount of product shipped from a platform $p \in P$ to terminal $t \in T(p)$ at a day d in a solution $s \in S_p$, and let $C_{p,s}$ be the cost of a solution $s \in S_p$ for a platform $p \in P$. Binary variables $\lambda_{p,s}$ indicate if a solution $s \in S_p$ for a platform $p \in P$ is used. The reformulation obtained after applying the Dantzig-Wolfe decomposition, the so called Dantzig-Wolfe Master ($\mathcal{DWM}$), is the following:

$$(\mathcal{DWM}) \min \sum_{p \in P} \sum_{s \in S_p} C_{p,s} \lambda_{p,s} \tag{24}$$

subject to

$$\sum_{s \in S_p} \lambda_{p,s} = 1 \quad \forall p \in P \tag{25}$$

$$\sum_{p \in P} \sum_{s \in S_p} \sum_{\tau=1}^{d} V_{p,s}^{t,\tau - D_{p,t}} \lambda_{p,s} \geq \left\lceil \frac{AC_{t,d}}{L_t} \right\rceil L_t \quad \forall t \in T, d \in \{1, ..., D\} \tag{26}$$

$$\sum_{p \in P} \sum_{s \in S_p} \sum_{\tau=1}^{d} V_{p,s}^{t,\tau - D_{p,t}} \lambda_{p,s} \leq \left\lfloor \frac{CAP_t + AC_{t,d}}{L_t} \right\rfloor L_t \quad \forall t \in T, d \in \{1, ..., D\} \tag{27}$$

$$\lambda_{p,s} \in \{0, 1\} \quad \forall p \in P, s \in S_p. \tag{28}$$

The objective function (24) aims to minimize the sum of transportation costs. Constraints (25) ensure that exactly one solution $s \in S_p$ variable is chosen for each platform $p \in P$. Constraints (26) avoid the inventory level at each terminal to be less than zero. Constraints (27) ensure that the maximum inventory capacity of each terminal will not be exceeded. Finally, constraints (28) define the λ variables as binaries.

3.2 Pricing subproblem

Let π_p be the vector of dual variables associated to constraints (25), $\theta_{t,d}$ be the vector of dual variables associated to constraints (26), and $\sigma_{t,d}$ be the vector of dual variables associated to constraints (27). The binary variables $z_{p,t}^{c,d}$ indicate wether a platform $p \in P$ sends a tanker of class $c \in C(p)$ to a terminal $t \in T(p)$ at a day d. Thus, for each platform $p \in P$, a pricing subproblem is defined as follows.

$$(\mathcal{SP}_p) \min \sum_{t \in T(p)} \sum_{c \in C(p)} \sum_{d=1}^{D} (2F_c D_{p,t} - V_c(\theta_{t,d+D_{p,t}} + \sigma_{t,d+D_{p,t}})) z_{p,t}^{c,d} - \pi_p \tag{29}$$

subject to

$$\sum_{t \in T(p)} \sum_{\tau=1}^{d} \sum_{c \in C(p)} z_{p,t}^{c,\tau} V_c \leq \left\lfloor \frac{AC_{p,d}}{L_p} \right\rfloor L_p \quad \forall d \in \{1, ..., D\} \tag{30}$$

$$\sum_{t\in T(p)}\sum_{\tau=1}^{d}\sum_{c\in C(p)} z_{p,t}^{c,\tau}V_c \geq \left\lceil \frac{AC_{p,d}-CAP_p}{L_p} \right\rceil L_p \quad \forall d \in \{1,...,D\} \tag{31}$$

$$z_{p,t}^{c,d} \in \{0,1\} \quad \forall t \in T(p), c \in C(p), d \in \{1,...,D\}. \tag{32}$$

The purpose of the Pricing Subproblem is to generate a λ variable to be added to the Restricted Master Problem (RMP) with the lowest reduced cost, which is represented by the objective function (29). Constraints (30) avoid the inventory level at each platform to be less than zero. Constraints (31) ensure that the maximum inventory capacity at each platform will not be exceeded. At last, z variables are defined as binary by (32). Even though the Pricing Subproblem is presented as an Integer Linear Program, it can be solved more efficiently by dynamic programming.

3.3 Dynamic Programming

For the Dynamic Programming, let c_f and c_l be, respectively, the first and the last tanker class of C and consider the following definitions:

$$P_{1,p,c,d,e} = \{\alpha \in \{0,1\}^{|T|} | 0 \leq e + P_{p,d} - \textstyle\sum_{t\in T} V_c\alpha_t\}$$
$$P_{2,p,c,d,e} = \{\alpha \in \{0,1\}^{|T|} | 0 \leq e - \textstyle\sum_{t\in T} V_c\alpha_t\}$$
$$P_{3,p,c,d,e} = \{\alpha \in \{0,1\}^{|T|} | 0 \leq e - \textstyle\sum_{t\in T} V_c\alpha_t \leq CAP_p\}$$

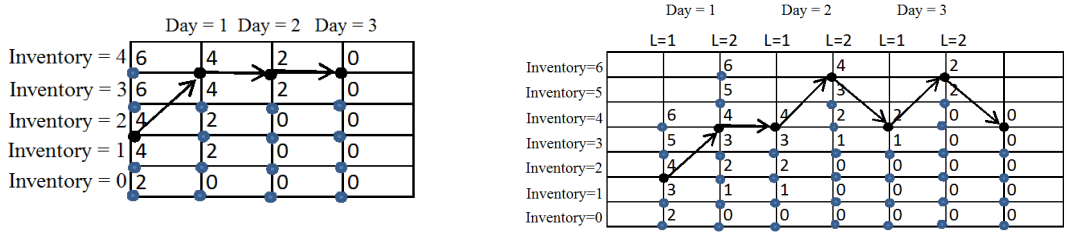
Let $C_{p,t}^{c,d} = 2F_cD_{p,t} - V_c(\theta_{t,d+D_{p,t}} + \sigma_{t,d+D_{p,t}})$. Let $Cost(d,c,e)$ be the cheapest cost from day d to the end of the time horizon, considering only the tanker classes $c,...,c_l$ at day d (and all classes for the previous days), and inventory level e at the beginning of day d. Define $Cost(d,c,e) = 0$ for $d = D+1$. Given this, the dynamic programming recursion is as follows:

$$\text{Cost}(d,c,e) = \begin{cases} \min\limits_{\alpha\in P_{1,p,c,d,e}} \left\{ Cost\left(d,c+1,e+P_{p,d}-\sum\limits_{t\in T}V_c\alpha_t\right) + \sum\limits_{t\in T} C_{p,t}^{c,d}\alpha_t \right\}, c = c_f \\ \min\limits_{\alpha\in P_{2,p,c,d,e}} \left\{ Cost\left(d,c+1,e-\sum\limits_{t\in T}V_c\alpha_t\right) + \sum\limits_{t\in T} C_{p,t}^{c,d}\alpha_t \right\}, c_f < c < c_l \\ \min\limits_{\alpha\in P_{3,p,c,d,e}} \left\{ Cost\left(d+1,1,e-\sum\limits_{t\in T}V_c\alpha_t\right) + \sum\limits_{t\in T} C_{p,t}^{c,d}\alpha_t \right\}, c = cl, \end{cases}$$

The optimal cost is $Cost(0,1,e_{p,0})$. The Dynamic Programming algorithm is pseudopolynomial. Its computational complexity depends on the maximum inventory capacity of the platforms. For the harder class of instances, its performance is quite superior to when a MIP is used. Figure 1a shows the simplest case, with just one lot size. Thus, $C = \{c_f\}$. The daily production is always integer. In this example, the inventory capacity of a platform is $CAP_p = 4$, the lot size is $V_1 = 2$, the production is the same for all days and is given by $P_{p,d} = 2$, $d \in \{1,2,3\}$. The initial inventory is 2 units. There is one terminal with route to all platforms. The daily production is added to the inventories between the days.

In Figure 1a, the costs for each inventory level and for each day are represented by the circles. At time $d = 0$, there is an initial inventory of 2 units and the lowest cost is $\text{Cost}(0,1,2) = 4$, which is also the optimal cost. From $d = 0$ to $d = 1$, there are two possibilities: (i) to send a lot ($\alpha_t = 1$), resulting in inventory of 2 units and cost 2; or (ii) do not send a lot ($\alpha_t = 0$), resulting in inventory of 4 units and cost 4. The second option is chosen. From $d = 1$ to $d = 2$, a lot is sent. The same happening from $d = 2$ to $d = 3$, resulting in an inventory of 4 units at the end of the time horizon.

Figure 1b shows an example with two lot sizes ($C = \{c_f, c_l\}$), but keeping the daily production still integer. There are two terminals with route to all platforms. Each lot size can be sent only once per day, for each terminal. In this example, the inventory capacity of the platform



(a) Dynamic programming with one lot size

(b) Dynamic programming with two lot sizes

Figure 1: Dynamic programming

is $CAP_p = 4$ units, the daily production is $P_{p,d} = 2$ units, $d \in \{1, 2, 3\}$, which is added between the first and the second lot size, and the initial inventory is 2 units. The first lot has size $V_1 = 1$ unit and costs 1.5. The second lot has size $V_2 = 2$ units and costs 2. Both can be sent to all terminals, with the same cost. If there are two or more terminals to send a lot size with different costs, the algorithm chooses always the cheapest terminal available. This happens because the cost of sending a lot depends on the transportation time. The economy of scale using bigger lot sizes does not increase the runtime. This is an advantage when compared to the MIP formulation for the pricing subproblem. The optimal solution is to send the second lot size at days $d = 2$ and $d = 3$, resulting in an inventory $e_3 = 4$ units at the end of the time horizon. The total cost is $\text{Cost}(0, 1, 2) = 4$.

For the case where the daily production is fractional at least one day, we need to do a preprocessing. We accumulate the daily productions, and if the accumulated value is fractional, we decrease the inventory capacity by one unit to have space for the fractional part. The adjusted daily production is the integer part of the accumulated production, not added yet to the inventory. Table 1 shows an example with fractional production where the inventory capacity has 6 units. The first column shows the daily original production, which is fractional and equal 2.75 for all days. The second column shows the accumulated production. Note that there is only one integer value. Third column shows the original capacity, 6 units. Fourth column shows the adjusted capacity, which is equal the original capacity if the accumulated production is integer, or equal the original capacity minus one, if the accumulated capacity is fractional. The fifth column shows the adjusted production, which is equal the integer part of the accumulated production not yet added to the inventory.

Table 1: Dealing with fractional production

Original production	Accumulated production	Original capacity	Adjusted capacity	Adjusted production
2.75	2.75	6	5	2
2.75	5.5	6	5	3
2.75	8.25	6	5	3
2.75	11	6	6	3
2.75	13.75	6	5	2

3.4 Column Generation-Based Heuristic Procedure

The main goal of the heuristic procedure presented in this section is to find a feasible solution in a smaller amount of time when compared to the heuristic procedures embedded in the commercial MIP solver. As the $\mathcal{RCAS}$ formulation has some difficulty to find feasible solutions at the beginning of the branch-and-bound algorithm for some instances, we implement a constructive heuristic using column generation to choose only some z variables with good chance to appear

in the optimal solution. This reduction in the number of variables allows the branch-and-bound algorithm to find fast a feasible solution, or prove that no solution can be found with these variables. Such Column Generation based heuristic is shown in Algorithm 1. First, the linear relaxation of the $\mathcal{DWM}$ reformulation is solved by column generation (line 1). Next, the reduced costs of all columns present in the final RMP are calculated using the values of the dual variables in the last iteration of the column generation algorithm (line 2). As these columns must have non-negative reduced costs, we then sort such columns in ascending order according to their reduced costs (line 3). The parameters min and max are initialized in line 4. In line 5, the parameter k is initialized. This parameter indicates that the kth first columns of the final RMP must be converted into variables of $\mathcal{RCAS}$. At each iteration of the main loop (lines 6 – 20), the $\mathcal{RCAS}$ formulation is populated with only the z variables corresponding to the k columns with less positive reduced costs in the final RMP (line 7). The $\mathcal{RCAS}$ formulation run for up to five seconds (line 8). If the obtained solution is proved to be optimal (line 9), we keep the solution if it improves the best found so far and increase k by ten. If a feasible solution is found but optimality was not proved (line 12), we keep the solution if it improves the best found so far but finish the loop. Lastly, if the solver can not find any feasible solution in 5 seconds (line 16), the loop stops. Note that if the problem is infeasible, the algorithm adds ten to k and try again. Then we run the $\mathcal{RCAS}$ with the best solution found as input (line 21). If no solution is found, the $\mathcal{RCAS}$ is run without any initial feasible solution.

Algorithm 1 Column Generation based heuristic

1: Run column generation to solve $\mathcal{DWM}$
2: Calculate the reduced cost of the columns using the dual variables of the last iteration
3: Sort the columns according to their reduced costs
4: $min \leftarrow 30$, $max \leftarrow 150$
5: $k \leftarrow min$
6: **while** $k \leq max$ **do**
7: Convert k columns of $\mathcal{DWM}$ with less positive reduced costs into variables of $\mathcal{RCAS}$
8: Run $\mathcal{RCAS}$ for up to five seconds
9: **if** optimal solution found **then**
10: Keep the solution if the best found
11: **end if**
12: **if** feasible solution found **then**
13: Keep the solution if the best found
14: Exit while
15: **end if**
16: **if** no solution found **then**
17: Exit while
18: **end if**
19: $k \leftarrow k + 10$
20: **end while**
21: Run $\mathcal{RCAS}$ for the remaining time using the best solution found

4 Computational results

In this section we show the results obtained by the described formulations using the instances proposed in Rocha *et al*. (2011) and Aizemberg *et al*. (2012). All the tests, including those for the formulation with the best performance proposed by Rocha *et al*. (2011) (*HullRel*) and by Aizemberg *et al*. (2012), were run in a PC with an Intel Core 2 Quad Q8300 2.50 GHz CPU, 2 GB RAM under Windows 7 OS using ILOG CPLEX 12.1. All instance classes have 25 instances and all instances have 9 platforms and 2 terminals. The first instance class has one lot size per route

(medium) and the second has two lot sizes per route (hard). In those two classes, some routes are missing. The third instance class has five lot sizes per route and all platforms are allowed to send product to all terminals (harder).

The results for the medium and hard classes of instances are shown separately from those for the harder class of instances. The tables contain, for each formulation, the average results obtained. Among the results, two performance measures are shown, given by:

$$gap_{LB} = 100 \times \frac{UB_b - LB_f}{UB_b}, \qquad (33) \qquad gap_{UB} = 100 \times \frac{UB_f - UB_b}{UB_b}, \qquad (34)$$

where UB_b is the best integer feasible solution found among all formulations, LB_f is the lower bound for a given formulation, and UB_f is the best integer feasible solution found by a given formulation.

The results obtained for the medium and hard classes of instances are shown in Tables 2 and 3, respectively. In the tables, the column **Formulation** shows the formulations tested. The formulation $\mathcal{RCAR}$ is the Rounded Cascading Accumulated in Routes formulation and the $\mathcal{RCAS}$ is the Rounded Cascading Accumulated in Sites formulation. Column **Time (s)** shows the average execution time in seconds for each class of instances. For all tests performed, the maximum execution time allowed for each instance is 720 seconds. Column $\mathbf{gap_{LB}}$ **root (%)** shows the average gap_{LB} at the root node, calculated using the gap_{LB} of each instance given by (33) considering LB_f as the lower bound at the root node. Column **final** $\mathbf{gap_{LB}}$ **(%)** contains the average gap_{LB} at the end of the execution time of each instance and it is also calculated by (33), where LB_f is the lower bound at the end of the execution time of each instance. Column $\mathbf{gap_{UB}}$ **(%)** shows the average gap_{UB} obtained at the end of executions and calculated by (34) for each instance. Column **No. of nodes** contains the average number of nodes solved at the branch-and-bound algorithm. Finally, column **Instances solved** shows the number of instances of each class solved to optimality.

Table 2: Average results and number of instances solved for the mathematical formulations and results taken from Rocha *et al*. (2011) for the medium class of instances

Formulation	Time (s)	gap_{LB} root (%)	final gap_{LB} (%)	gap_{UB} (%)	No. of nodes	Instances solved
$\mathcal{RCAS}$	1.28	0.23	0.00	0.00	3405	all
$\mathcal{RCAR}$ (*Aizemberg et al*)	30.61	0.18	0.00	0.00	146637	24
HullRel (*Rocha et al*)	351.53	–	–	0.04	–	13

Table 3: Average results and number of instances solved for the mathematical formulations and results taken from Rocha *et al*. (2011) for the hard class of instances

Formulation	Time (s)	gap_{LB} root (%)	final gap_{LB} (%)	gap_{UB} (%)	No. of nodes	Instances solved
$\mathcal{RCAS}$	98.21	1.30	0.77	0.00	32761	22
$\mathcal{RCAR}$ (*Aizemberg et al*)	102.94	1.47	0.79	0.03	53669	22
HullRel (*Rocha et al*)	691.40	–	–	1.24	–	1

In Tables 2 and 3 one can observe that the new formulation outperforms the literature results reported in Rocha *et al*. (2011) and Aizemberg *et al*. (2012). Even though the *HullRel* has a low average gap_{UB}, this formulation was capable of solving only thirteen instances of the medium class to optimality, and one of the hard class. This fact impacts in the average execution time, which is close to the maximum allowed, for the hard class of instances.

Comparing the $\mathcal{RCAR}$ with $\mathcal{RCAS}$, we can note that accumulating the lot sizes per site is better than per route. We believe that this happens because there are more routes than sites in all instances. In Table 2, we can see that the execution time of the $\mathcal{RCAS}$ is almost zero, and this formulation solved to optimality all the 25 instances. The $\mathcal{RCAR}$ formulation has a very low execution time too, but could not solve all 25 instances. The number of nodes visited of the $\mathcal{RCAS}$ formulation is much smaller than $\mathcal{RCAR}$. In Table 3, we can see that the $\mathcal{RCAS}$ is still better than the $\mathcal{RCAR}$, but the difference is small. Both have small execution times and gap_{UB}, and solved 22 of 25 instances to optimality. We conclude that, for the instances used in this paper, $\mathcal{RCAS}$ is the best formulation.

Table 4: Average results and number of instances solved for the $\mathcal{RCAS}$ formulation with and without the column generation-based heuristic for the harder class of instances generated

Formulation	Time (s)	gap_{LB} root (%)	gap_{LB} final (%)	gap_{UB} (%)	No. of nodes	Instances solved
$\mathcal{RCAR}$(*Aizemberg et al*)	637.64	8.79	7.16	–	14044	3
$\mathcal{RCAS}$	550.15	5.90	5.01	0.68	5935057	7
$\mathcal{RCAS} - CGH$	498.62	5.81	4.76	0.13	130949	10

We use the harder class of instances to compare the formulations with the best performance on the other classes, $\mathcal{RCAR}$ and $\mathcal{RCAS}$, and the $\mathcal{RCAS}$ preceded by the heuristic described in subsection 3.4 ($\mathcal{RCAS} - CGH$). We note that, for instances without feasible solution after run $\mathcal{RCAS}$ for 12 minutes, the heuristic has reduced chance to find a feasible solution. However, in case where the model finds some solution within the 12 minutes, the heuristic usually finds an improved solution to use as input to the $\mathcal{RCAS}$ formulation very quickly.

The results obtained for the harder class of instances are shown in Table 4. Comparing $\mathcal{RCAS}$ to $\mathcal{RCAR}$, we can see a good improvement. $\mathcal{RCAS}$ proved the optimality of 7 instances, against 3 of $\mathcal{RCAR}$. The average final gap_{LB} of $\mathcal{RCAS}$ is 30% smaller. Comparing the $\mathcal{RCAS}$ with and without the heuristic, we can see a good improvement when the heuristic is used, solving to optimality 10 of the 25 instances, against 7 without heuristic. The average gap_{UB} shows that, as expected, the heuristic allows the formulation to find better feasible solutions. The average Time of $\mathcal{RCAS} - CGH$ include the heuristic runtime. Considering only the heuristic runtime, the average time is 11.53 seconds, with a minimum of 7 seconds and a maximum of 16.78 seconds. The root gap_{LB}, final gap_{LB} and the gap_{UB} in Table 4 are compared with the best solution found for each instance. To find optimal or near optimal solution of all instances of the harder class (to be used as the UB_b value), we run each instance for many hours, using as cutoff the best solution found in previous tests.

5 Conclusions

This work presented a modeling study for the oil transportation problem. For this problem, a new formulation was devised: the Rounded Cascading Accumulated in Sites Formulation ($\mathcal{RCAS}$). This formulation was compared to two formulations from literature, the Rounded Cascading Accumulated in Routes Formulation ($\mathcal{RCAR}$) (Aizemberg *et al.*, 2012) and the Knapsack Cascading ($\mathcal{KC}$) (Rocha *et al.*, 2011), achieving a better performance for all three classes of instances taken from literature. For the $\mathcal{RCAS}$, we construct a column generation-based heuristic that improve the formulation performance.

References

Aizemberg, L., Uchoa, E., Pessoa, A., Kramer, H. H., Coutinho, R. and Paula Junior, U. C. Formulações para o problema do transporte de derivados de petróleo. *Anais do XLIV SBPO 2012*, Rio de Janeiro–RJ, Brasil. In Portuguese, 2012.

Aizemberg, L., Uchoa, E., Pessoa, A., Rocha, R., Coutinho, R. and Paula Junior, U. C. Modelo de otimização para o problema do transporte de derivados de petróleo com busca local por MIP e simulação. *Anais do XLIII SBPO 2011*, Ubatuba–SP, Brasil. In Portuguese, 2011.

Bahl, H. C. (1983), Column generation based heuristic algorithm for multi-item scheduling. *IIE Transactions*, v. 15, n. 2, p. 136–141.

Banaszewski, R. F., Tacla, C. A., Pereira, F. R., Arruda, L. V. and Enembreck, F. Planning transport of crude oil derivatives with simultaneous auctions. *2010 IEEE International Conference on Systems Man and Cybernetics (SMC)*, p. 2820–2827. IEEE, 2010.

Boschetto, S. N., Magatão, L., Brondani, W. M., Neves-Jr, F., Arruda, L. V. R., Barbosa-Póvoa, A. P. F. D. and Relvas, S. (2010), An operational scheduling model to product distribution through a pipeline network. *Industrial & Engineering Chemistry Research*, v. 49, n. 12, p. 5661–5682.

Choi, E. and Tcha, D.-W. (2007), A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, v. 34, n. 7, p. 2080 – 2095.

Dantzig, G. B. and Wolfe, P. (1960), Decomposition principle for linear programs. *Operations research*, v. 8, n. 1, p. 101–111.

Furini, F., Malaguti, E., Durain, R. M., Persiani, A. and Toth, P. (2012), A column generation heuristic for the two-dimensional two-staged guillotine cutting stock problem with multiple stock size. *European Journal of Operational Research*, v. 218, n. 1, p. 251 – 260.

Joncour, C., Michel, S., Sadykov, R., Sverdlov, D. and Vanderbeck, F. (2010), Column generation based primal heuristics. *Electronic Notes in Discrete Mathematics*, v. 36, n. 0, p. 695 – 702.

Magatão, L., Arruda, L. V. R., Neves, F. et al. (2004), A mixed integer programming approach for scheduling commodities in a pipeline. *Computers & chemical engineering*, v. 28, n. 1-2, p. 171–185.

Moccia, L., Cordeau, J.-F., Monaco, M. F. and Sammarra, M. (2009), A column generation heuristic for a dynamic generalized assignment problem. *Computers & Operations Research*, v. 36, n. 9, p. 2670 – 2681.

Mourgaya, M. and Vanderbeck, F. (2007), Column generation based heuristic for tactical planning in multi-period vehicle routing. *European Journal of Operational Research*, v. 183, n. 3, p. 1028 – 1041.

Rocha, R., Grossmann, I. E. and Poggi de Aragão, M. V. S. (2011), Cascading knapsack inequalities: reformulation of a crude oil distribution problem. *Annals of Operations Research*, v. , p. 1–18.

Velez, S. and Maravelias, C. T. (2013), Reformulations and branching methods for mixed-integer programming chemical production scheduling models. *Industrial & Engineering Chemistry Research*, v. 52, n. 10, p. 3832 – 3841.

Yavuz, B., Aras, N., Kuban, I. and Ersoy, C. (2010), A column generation based heuristic for sensor placement, activity scheduling and data routing in wireless sensor networks. *European Journal of Operational Research*, v. 207, n. 2, p. 1014 – 1026.