ADCHEM 2006

International Symposium on Advanced Control of Chemical Processes
Gramado, Brazil – April 2-5, 2006

THE FACILITY LOCATION PROBLEM: MODEL, ALGORITHM, AND APPLICATION TO COMPRESSOR ALLOCATION

Eduardo Camponogara [*,1],
Melissa Pereira de Castro [*,2],
Agustinho Plucenio [*,2]

[*] *Department of Automation and Systems, Federal University of Santa Catarina, Cx.P. 476, 88040-900 Florianópolis, SC, Brazil*

Abstract: Unlike other problems found in the oil industry, the allocation of compressors to gas-lifted oil wells lacks formal models and algorithms capable of delivering globally optimal assignments. To this end, this paper casts the compressor allocation problem as a facility location problem. Owing to the special structure of the target problem, an efficient (polynomial time) algorithm is conceived by applying the framework of dynamic programming. An example illustrates the inner-workings of the algorithm. Connections between the polyhedron of feasible solutions and integer-programming theory are established.

1. MOTIVATION

The prospection for oil reserves, the recovery of oil, and the distillation into final products are complex processes that rely heavily on technology. Since the inception of the oil industry, production and decision processes have been automated, in part to cut costs and improve efficiency in response to the mounting pressure from competitive markets, but also due to the continuous transformation of scientific advances into technology (Jahn et al., 2003). For instance, a plan to recover oil from a reservoir takes into account geological studies of the formation, data from seismic analysis, and predictions from simulators. A plan will pinpoint the locations for drilling, along with the technology and apparatus to artificially lift hydrocarbons from the reservoir to surface facilities, where they are separated into oil, water, and gas. Among the artificial lifting techniques (Nishikiori et al., 1995; Buitrago et al., 1996), continuous gas-lift is a favored technique for its relatively low installation and maintenance costs, wide range of operating conditions, and robustness. Continuous gas-lift works by injecting high pressure gas at the bottom of the production tubing to reduce the weight of the oil column, thereby forcing the flow of fluid to surface facilities. In view of the complexity of planning and operating the production processes of an oil field as a whole, the overall task is invariably broken down to smaller, tractable sub-problems that can be solved more efficiently with the aid of models and algorithms. In particular, two relevant sub-problems are lift-gas allocation and compressor scheduling.

Lift-gas allocation concerns the distribution of a limited gas rate to the oil wells, while respecting lower and upper bounds for gas injection at each well, maximum compressing capacity, and surface facility constraints with the aim of minimizing production costs. A number of models and algorithms appeared in the literature (Kanu et al., 1981; Buitrago et al., 1996; Alarcón et al., 2002), but more recently globally optimal algorithms

[1] Partially supported by CNPq
[2] Supported by the Brazilian Agency of Petroleum (ANP)

have been developed to handle discrete decisions and the non-linear nature of well performance curves (Camponogara and Nakashima, 2006b,a).

On the other hand, compressor scheduling concerns the allocation of compressors to meet the pressure levels set down by the (oil field) recovery plan for each oil well. Setting up compressor facilities to precisely meet the pressure needs of each well is not only inconvenient, but the installation costs are also excessive. In a typical situation, the decision-maker has to trade-off the installation costs and energy-loss costs incurred by reducing the pressure yielded by the scheduled compressor to the well's level. The task of deciding which compression levels (*facilities*) will be installed and how they will supply lift-gas to the wells (*clients*) so as to minimize the overall cost gives rise to the compressor allocation problem (CAP). Unlike lift-gas allocation, the compressor allocation problem has received little attention, lacking formal models and provably optimal algorithms. This works is a first attempt to bridge the gap between industry and academia, having the aim of formally stating the problem, developing a baseline algorithm, and pointing out directions for further research.

2. PROBLEM FORMULATION

Roughly speaking, in the facility location problem one has to decide upon the location of facilities and how these facilities supply clients' demands, while minimizing the aggregated cost incurred by installing facilities and transporting goods to clients (Wolsey, 1998; Cornuejols et al., 1977). $N = \{1, \ldots, n\}$ is the set of potential sites for facilities and $M = \{1, \ldots, m\}$ is the set of clients. A fixed cost a_j is charged to set up facility j, while c_{ij} is the cost to fulfill the whole demand of client i from facility j. Because clients have special needs, a client i can be serviced only by a subset $N_i \subseteq N$ of the facilities. Let $M_j = \{i \in M : j \in N_i\}$ be the subset of potential clients of facility j. The problem can be cast in mathematical programming as follows:

$$P : \text{Minimize } J = \sum_{j=1}^{n} a_j y_j + \sum_{i=1}^{m} \sum_{j \in N_i} c_{ij} x_{ij} \quad (1a)$$

Subject to :

$$x_{ij} \leqslant y_j, \ i \in M, \ j \in N_i \quad (1b)$$

$$\sum_{j \in N_i} x_{ij} = 1, \ i \in M \quad (1c)$$

$$y_j \in \{0, 1\}, \ j \in N \quad (1d)$$

$$x_{ij} \in \{0, 1\}, \ i \in M, \ j \in N_i \quad (1e)$$

Despite its clear statement and simple model, the computational solution of the facility location problem is intrinsically hard, that is to say, a polynomial-time algorithm for P would entail solving the class of NP-Hard problems in polynomial time (Garey and Johnson, 1979). One such problem is set covering (Nemhauser and Wolsey, 1988): given a set S, a family $\{S_j\}_{j=1}^{n}$ of subsets of S, and the cost c_j for each S_j, the problem rests on finding a subset $U \subseteq \{1, \ldots, n\}$ that minimizes $\sum_{j \in U} c_j$ and such that $\cup_{j \in U} S_j = S$ (i.e., the selected subsets induce a cover of S), which gives the name to the problem. It is a straightforward exercise to reduce set covering to the facility location problem in polynomial time and space—just associate each facility j with a subset S_j, model each client i as an element of S, equate M_j to S_j and define N_i accordingly, set a_j as c_j, and set $c_{ij} = 0$ for all i and j; the optimal solution to the facility location problem is precisely a minimum cost cover for S.

Not surprisingly, the target application of compressor allocation can be framed as a facility location problem, with the facilities representing the potential pressure levels for compressor pools, whereas the clients represent the oil wells. In this scenario, compressing station j yields lift-gas flow at pressure p_j^c. The problem arises from the allocation of these pressure levels to the oil wells so as to meet the recovery plan of the oil field, according to which lift-gas should be injected in well i at pressure p_i^w. Henceforth, this problem will be referred to as *compressor allocation problem* (CAP). Since pressure can only be reduced without energy gain, but invariably incurring energy losses, pressure p_j^c must be greater than or equal to pressure p_i^w for compressor j to be allocated to well i.

Assumption 1. An instance of CAP satisfies the properties:

(1) $p_1^c > p_2^c > \ldots > p_n^c$;
(2) $p_1^w \geqslant p_2^w \geqslant \ldots \geqslant p_m^w$;
(3) $p_1^c \geqslant p_1^w$, implying that demands from wells can be fulfilled;
(4) for all $i \in M$ and $j, k \in N_i, j < k$, $c_{ij} \geqslant c_{ik}$, expressing that the energy-loss cost to drop compressor pressure p_j^c to p_i^w, c_{ij}, is not lower than dropping from p_k^c, c_{ik}, because $p_j^c > p_k^c$.

The assumptions made above are typical of instances of the compressor allocation problem and, therefore, do not limit the developments hereafter but rather bring out the intrinsic structure of the problem. This structure can be exploited to develop specially-tailored algorithms that are far more efficient than general-purpose algorithms for the facility location problem. Actually, we will design a polynomial-time algorithm for facility location problems arising from instances of the compressor allocation problem.

3. DYNAMIC PROGRAMMING ALGORITHM

3.1 Background

Dynamic programming (DP) is a framework for breaking up problems in a set of smaller, easier to solve sub-problems arranged in a sequence (Bertsekas, 1995). The decisions are made in stages, each regarding a decision at the present time and its future consequences.

In a deterministic situation, the consequences are fully predictable whereas, in a stochastic situation, the outcomes are ruled by chance and modeled by random variables. The goal is to reach decisions that minimize the cumulative cost stretching from the current time until the end of the time horizon. Because each decision influences future outcomes, the decision-maker has to balance the desire of low costs at the current time and the undesirability of high future costs. Dynamic programming balances the trade-off between current and future costs: at each stage, the decisions are ranked according to the sum of the present cost and future costs, assuming optimal decision making over the subsequent stages, also known as *cost-to-go* (from the next state until termination).

For the stochastic scenario, the costs are expected values over the distributions of the random variables. DP has been successfully applied to a wide range of problems, including optimal control and discrete optimization problems.

In the domain of control theory, the typical task is to minimize system's state distance from a desired trajectory while factoring in the control-action costs and balancing the present and future costs. The system's state evolves over time according to discrete-time dynamic equations that might have random variables, such as the number of service requests arriving at a shop, random disturbances, or the number of units ordered from clients. Because an optimal control policy can only be determined after the observation of the random variables, the algorithmic solution is structured backward, working from the terminal stage toward the current time.

In the domain of optimization, the typical task is not unlike in control theory, where the system's state might represent the quantity of resources available at the moment, such as the remaining budget to pay for tolls to reach the destination or the remaining capacity in a knapsack. The applications in discrete optimization abound, including resource-constrained shortest-path problems, knapsack-like problems, and string matching (Cormen et al., 1990; Wolsey, 1998).

3.2 Recursive Formulation

Here, we take advantage of the structure of CAP, as formalized in Assumption 1, to break P into a sequence $\{P_m(s_m), P_{m-1}(s_{m-1}), \ldots, P_1(s_1)\}$ of sub-problems. $P_i(s_i)$ is a restricted form of P in which one has to allocate facilities to a subset of clients, namely $\{i, i+1, \ldots, m\}$, given that the facilities in $\{1, \ldots, s_i - 1\}$ are unavailable, facility s_m has been installed, and the installation of the facilities from $\{s_i+1, \ldots, n\}$ are to be decided. Let $J_i(s_i)$ be the value of an optimal solution to $P_i(s_i)$. Parameter s_i acts as the "*state of the system*" at stage i. Note that $P_1(0)$ is equivalent to P.

Rather than solving $\{P_i(s_i)\}$, we recursively solve the sequence $\{\hat{P}_i(s_i)\}$ given below, taking advantage of the problem structure to reduce the search for an optimal solution dramatically. The first problem of the sequence $\{\hat{P}_i(s_i)\}$ is expressed in mathematical programming as:

$$\hat{P}_m(s_m):$$

$$\hat{J}_m(s_m) = \text{Min} \sum_{j \in N_m^+} a_j y_j + \sum_{j \in N_m^-} c_{mj} x_{mj} \quad (2a)$$

S. to :

$$x_{mj} \leqslant y_j, \ j \in N_m^+ \quad (2b)$$

$$\sum_{j \in N_m^-} x_{mj} = 1 \quad (2c)$$

$$y_j \in \{0,1\}, \ j \in N_m^+ \quad (2d)$$

$$x_{mj} \in \{0,1\}, \ j \in N_m^- \quad (2e)$$

where:

- $N_m^+ = N_m \cap \{s_m + 1, \ldots, n\}$;
- $N_m^- = N_m \cap \{s_m, \ldots, n\}$; and
- $s_m \in \{0\} \cup N$ is the index of the lowest pressure level already installed.

In case $\{s_m, s_m + 1, \ldots, n\} \cap N_m = \emptyset$, $\hat{J}_m(s_m)$ becomes $+\infty$ as neither facility level s_m nor its succeeding levels can service client m. In case $s_m = 0$, one can choose the facility that induces the lowest combined cost of installation and transportation, that is, $\min\{a_j + c_{mj} : j \in N_m\}$.

For the remaining stages, the problem at stage i accounts for the cost to service client i and the clients in $\{i+1, \ldots, m\}$ (*cost-to-go* from the current stage and state to the terminal state), given that facilities $\{1, \ldots, s_i - 1\}$ are decommissioned and facility s_i is in service (if $s_i > 0$). The problem encompassing the decisions from stage i till termination is cast as follows:

$\hat{P}_i(s_i)$:

$$\hat{J}_i(s_i) = \text{Min} \sum_{j \in N_i^+} a_j y_j + \sum_{j \in N_i^-} c_{ij} x_{ij} + \hat{J}_{i+1}(s_{i+1}) \quad (3a)$$

S. to :

$$x_{ij} \leqslant y_j,\ j \in N_i^+ \quad (3b)$$

$$\sum_{j \in N_i^-} x_{ij} = 1 \quad (3c)$$

$$s_{i+1} = \max\{s_i, j : y_j = 1\} \quad (3d)$$

$$y_j \in \{0,1\},\ j \in N_i^+ \quad (3e)$$

$$x_{ij} \in \{0,1\},\ j \in N_i^- \quad (3f)$$

The purpose and interpretation of $s_i \in \{0\} \cup N$ is the same as in $\hat{P}_m(s_m)$. In equation (3d), if no facility is installed at stage i then s_{i+1} takes on value s_i, or else s_{i+1} becomes the index of the facility that has been installed. In the event of $\hat{P}_i(s_i)$ being infeasible, which occurs when $\{s_i, s_i+1, \ldots, n\} \cap N_i = \emptyset$, then $\hat{J}_i(s_i)$ becomes $+\infty$.

An effective DP algorithm can be designed to solve $\hat{P}_1(0)$ by recursively solving the problem family $\{\hat{P}_i(s_i)\}$ in an appropriate sequence, and skipping the instances that are clearly infeasible.

Dynamic Programming Algorithm

For $s_m = 0, \ldots, n$ do
  Solve $\hat{P}_m(s_m)$ to compute $\hat{J}_m(s_m)$
  Let $(x_m(s_m), y_m(s_m))$ be a solution to
    $\hat{P}_m(s_m)$ if $\hat{J}_m(s_m) < +\infty$, that is,
    $x_m(s_m) = (m, j)$ for which $x_{mj} = 1$ and
    $y_m(s_m) = 0$ if $j = s_m$ or else $y_m(s_m) = j$
For $i = m-1, \ldots, 1$ do
  For $s_i = 0, \ldots, n$ do
    Solve $\hat{P}_i(s_i)$ to compute $\hat{J}_i(s_i)$
    Let $(x_i(s_i), y_i(s_i))$ be the solution to
      $\hat{P}_i(s_i)$ which can be obtained as
      delineated above

Let $\hat{J}_i = \{\hat{J}_i(s_i) : s_i \in \{0\} \cup N_i\}$ be the table with the values of solutions to all the sub-problems at stage i, with the exception of the infeasible ones. The algorithm works by computing $\hat{J}_m$, $\hat{J}_{m-1}$, and so forth until reaching $\hat{J}_1$.

Of course, the decision maker needs the optimal solution, not only the optimal values. The algorithm outlined above bookkeeps the optimal decisions in tables $X_i = \{x_i(s_i) : s_i \in \{0\} \cup N_i\}$ and $Y_i = \{y_i(s_i)\}$, even though the decisions could be inferred from the tables $\hat{J}_i$, for the sake of simplicity. At termination, the optimal solution can be produced by the algorithm given below.

Optimal Solution Computation

Let y be a list of facilities to be installed
Let x be a list of client-facility pairs
$y \leftarrow \emptyset$, $x \leftarrow \emptyset$, and $s_1 \leftarrow 0$
For $i = 1, \ldots, m$ do
  $(i', j') \leftarrow x_i(s_i)$
  $x \leftarrow x \cup \{(i', j')\}$
  If $j' = s_i$
    then $s_{i+1} = s_i$
    else $s_{i+1} = j'$
      $y \leftarrow y \cup \{j'\}$

The procedure above outputs in y the indexes of the facilities to be installed and, in x, which facilities will be servicing which clients. The correctness of the dynamic programming is established below by showing that the recursive formulation $\{\hat{P}_i(s_i)\}$, given in (2a)–(2e) and (3a)–(3f), is equivalent to $\{P_i(s_i)\}$.

Lemma 1. $\hat{J}_i(s_i) = J_i(s_i)$ if Assumption 1 holds.

Proof: (By induction in i) For the basis, $i = m$, $\hat{P}_i(s_i)$ is obviously equivalent to $P_i(s_i)$ because only the facilities from N_m^+ may be installed, implying that $\hat{J}_m(s_m) = J_m(s_m)$.

For the induction step, $i < m$, suppose facility s_{i+1} services client i, where s_{i+1} must belong to N_i^-. Because $c_{kj} \geqslant c_{ks_{i+1}}$ for each $k \in \{i+1, \ldots, m\}$ and all $j \in N_k \cap \{s_i, \ldots, s_{i+1} - 1\}$ (refer to Assumption 1), and $a_j \geq 0$ for all j, the facilities in the set $N_k \cap \{s_i+1, \ldots, s_{i+1}-1\}$ do not need to be installed in an optimal solution. Consequently, $J_i(s_i) = c_{i,s_{i+1}} + J_{i+1}(s_{i+1}) = c_{i,s_{i+1}} + \hat{J}_{i+1}(s_{i+1}) = \hat{J}_i(s_i)$ by induction hypothesis if $s_{i+1} = s_i$ and, similarly, $J_i(s_i) = a_{s_{i+1}} + c_{i,s_{i+1}} + J_{i+1}(s_{i+1}) = a_{s_{i+1}} + c_{i,s_{i+1}} + \hat{J}_{i+1}(s_{i+1}) = \hat{J}_i(s_i)$ by induction hypothesis if $s_{i+1} > s_i$, completing the demonstration. ■

Corollary 1. $\hat{P}_1(0)$ is equivalent to P and $\hat{J}_1(0) = J$ is the value of an optimal solution.

Corollary 2. P can be solved in polynomial time.

Proof: The DP algorithm performs $\Theta(mn)$ steps each taking $O(n)$ steps to solve $\hat{P}_i(s_i)$. Thus, the algorithm runs in $O(mn^2) \in O(\max\{m, n\}^3)$ time and its memory usage is $\Theta(mn)$. ■

3.3 An Illustrative Example

The purpose of the material herein is to crystallize the concepts and illustrate the DP algorithm in a simple context. The scenario consists of $n = 4$ pressure levels (*facilities*) and $m = 4$ oil wells that

Table 1. Compressor and well data

Compressors				Oil wells		
j	p_j^c	a_j	M_j	i	p_i^w	N_i
1	10	8	$\{1,2,3,4\}$	1	9	$\{1\}$
2	8	6	$\{2,3,4\}$	2	8	$\{1,2\}$
3	6	10	$\{4\}$	3	7	$\{1,2\}$
4	4	4	$\{4\}$	4	3	$\{1,2,3,4\}$

Table 2. Energy loss costs

Energy loss cost: c_{ij}				
$i\backslash j$	1	2	3	4
1	8			
2	6	4		
3	10	8		
4	6	4	3	1

Table 3. Dynamic programming algorithm: sub-problem objective values

	$\hat{J}_i(s_i)$				
$i\backslash s_i$	0	1	2	3	4
1	37	29			
2	22	21	16		
3	18	15	12		
4	5	5	4	3	1

are operated via continuous gas-lift (*clients*). The pressure output of the compressors, the pressure demand of the wells, and the costs to install banks of compressors appear in Table 1. The penalties due to the energy loss resulting from pressure reduction are depicted in Table 2. This data comprises an instance of CAP.

3.4 Applying the DP Algorithm

The application of the dynamic programming algorithm to the instance given above yields the set $\hat{J} = \{\hat{J}_i : i \in M\}$ depicted in Table 3. The missing entries of the table have value $+\infty$, the default value for an infeasible sub-problem. Notice that $\hat{J}_4(s_4)$ is feasible for all s_4 given that client 4 can be serviced by all facilities. However, $\hat{J}_2(s_2)$ and $\hat{J}_3(s_3)$ take on value $+\infty$ when $s_2, s_3 \in \{3,4\}$ because neither facility 3 nor facility 4 can supply clients 2 and 3. The DP algorithm fills in the entries of client 4's row by solving $\hat{P}_m(s_m)$. The remaining lines are computed by solving $\hat{P}_i(s_i)$ for $i = 3,2,1$. The value of the optimal solution is therefore $\hat{J}_1(0) = 37$. Simultaneously, the solutions to these sub-problems are recorded in Table 4 for posterior retrieval. The optimal solution is obtained as follows: $x_1(0) = (1,1)$ and $y_1(0) = 1$ state that facility 1 should installed to service client 1; $x_2(1) = (2,1)$ and $y_2(1) = 0$ indicate that facility 1 services client 2; $x_3(1) = (3,1)$ and $y_3(1) = 0$ indicate that facility 1 services client 3; and $x_4(1) = (4,4)$ and $y_4(1) = 4$ indicate that facility 4 should be installed to supply client 4.

Table 4. Dynamic programming algorithm: optimal solutions

	$x_i(s_i)/y_i(s_i)$				
i	$s_i = 0$	1	2	3	4
1	$x_{11} = 1$	$x_{11} = 1$			
	$y_1 = 1$				
2	$x_{22} = 1$	$x_{21} = 1$	$x_{22} = 1$		
	$y_2 = 1$				
3	$x_{32} = 1$	$x_{31} = 1$	$x_{32} = 1$		
	$y_2 = 1$				
4	$x_{44} = 1$	$x_{44} = 1$	$x_{42} = 1$	$x_{43} = 1$	$x_{44} = 1$
	$y_4 = 1$	$y_4 = 1$			

3.5 Remarks

The particular structure of the compressor allocation problem allowed us to design an efficient, polynomial-time dynamic programming algorithm. A standing issue is whether or not this algorithm is optimal—put another way, the issue is whether $\Omega(mn^2)$ is a lower bound for the problem. If the problem data is given in matrix form, then the input size is of the order $\Theta(mn)$, and the issue is whether or not there exists an algorithm with running time between $\Theta(mn)$ and $\Theta(mn^2)$.

Another issue regards the design of a simpler, more efficient algorithm that would run in linear time regardless of the input size. The greedy algorithm, however, may fail to produce the optimal solution, as illustrated in the following scenario. Take the heuristic that scans the clients from 1 to m, deciding at each stage i whether to install a new facility or else service i with one of the existing facilities. The greedy choice would pick the least costly option. For the instance with $n = 2$, $a_1 = 10$, $a_2 = 8$, $N_1 = \{1\}$, $N_i = \{1,2\}$ for $i = 2,\ldots,m$, $c_{11} = 1$, $c_{i1} = 2$ and $c_{i2} = 0$ for $i = 2,\ldots,m$, the greedy heuristic fails when $m \geqslant 6$. It will install facility 1 and service all clients from this facility, incurring a total cost of $11+2(m-1)$, whereas the optimal solution installs facilities 1 and 2 with a total cost of 19. Likewise, the greedy heuristic that works in the opposite direction, from client m toward 1, may fail as well.

A feature of the dynamic programming algorithm is its simplicity: the algorithm is devoid of complex data structures and can be implemented in virtually any computer language. A second feature is the potential to use a parallel computer: because the computation of the values $\hat{J}_i(s_i)$ can be carried out in parallel for all $s_i \in \{0\} \cup N_i$, the algorithm may run in $O(mn)$ time when $\Theta(n)$ processors are available, thereby inducing maximum speed-up.

Let $\mathcal{P} = \{z \in \mathbb{R}^p : Az \leqslant b, 0 \leqslant z \leqslant 1\}$, $z = (x,y)$, be the polyhedron corresponding to a formulation of P, where $p = n + \sum_{j=1}^{n} |M_j|$ and matrix A consists of the constraints (1b) and (1c). Notice that P can be recast as $\max\{c^T z : z \in X\}$ where $X = \mathcal{P} \cap \mathbb{Z}^p$ and c is a suitable vector, this way giving $\mathcal{P}$ the status of a formulation.

From integer-programming theory (Nemhauser and Wolsey, 1988; Wolsey, 1998), follows that the convex hull of X, $conv(X)$, is also a polyhedron $\tilde{\mathcal{P}} = \{z \in \mathbb{R}^p : \tilde{A}z \leqslant \tilde{b}\} = conv(X)$ known as integer polyhedron. If $(\tilde{A}, \tilde{b})$ were known, one could instead solve $\max\{c^T z : \tilde{A}z \leqslant \tilde{b}\}$ with linear programming which, in turn, can be solved in polynomial time if the size of $(\tilde{A}, \tilde{b})$ is polynomial or if separation [3] can be performed in polynomial time. It so happens that certain problems have a formulation that is itself an integer polyhedron, that is, $\mathcal{P} = \tilde{\mathcal{P}}$. Examples include the matching and network-flow polyhedron. It has been established that the linear program $\max\{c^T z : Az \leqslant b, z \in \mathbb{R}^n_+\}$ has an integral solution for all integer vectors b for which it has a finite optimal value if and only if A is totally unimodular. A matrix A is totally unimodular (TU) if every square submatrix of A has determinant $+1, -1$, or 0. From a preliminary analysis, it appears that the matrix A arising from the constraints (1b) and (1c) is TU if the conditions of Assumption 1 hold, but this property is yet to be confirmed. An early result states that A is TU if, and only if, for every (square) *Eulerian* [4] submatrix B of A the sum of the entries of B divides by 4 (Camion, 1965). It seems plausible to use this result in trying to show that A is totally unimodular.

4. CONCLUSIONS AND FUTURE WORKS

The paper has formally stated the problem of allocating compressors to oil wells and framed it as a facility location problem. The special structure of the target problem enabled us to design an efficient, polynomial-time algorithm for an otherwise NP-Hard problem. A simple, but illustrative example helped to crystallize the concepts and the recursive principles upon which the algorithm was conceived. The polynomial-time solution of the compressor allocation problem raised the possibility of the constraint matrix being totally unimodular.

Future research will extend CAP to account for potential capacities of the compressors and time-dependent pressure demands for the oil wells.

ACKNOWLEDGMENTS

This research was funded by Agência Nacional de Petróleo (ANP) and Financiadora de Estudos e Projetos (FINEP), by means of a grant from Programa de Recursos Humanos da ANP para o Setor de Petróleo e Gás PRH-34 ANP/MCT.

[3] Let $S = \{\tilde{a}_i^T z \leqslant \tilde{b}_i, i = 1, \ldots, m\}$ be the set of constraints defined by $(\tilde{A}, \tilde{b})$. Given a fractional solution z, $z \notin \mathcal{P} \cap \mathbb{Z}^p$, the separation problem is to find a constraint i that is not respected by z, that is, $\tilde{a}_i^T z \not\leqslant \tilde{b}_i$.

[4] A matrix is said to be *Eulerian* if the sum of the entries of its lines and the sum of the entries of its columns are both even.

REFERENCES

G. A. Alarcón, C. F. Torres, and L. E. Gómez. Global optimization of gas allocation to a group of wells in artificial lift using nonlinear constrained programming. *ASME Journal of Energy Resources Technology*, 124(4):262–268, December 2002.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume I. Athena Scientific, Belmont, MA, 1995.

S. Buitrago, E. Rodríguez, and S. D. Espin. Global optimization techniques in gas allocation for continuous flow gas lift systems. In *Proc. SPE Gas Technology Conference*, pages 375–379, Calgary, Canada, 1996. Paper SPE 15616.

P. Camion. Characterization of totally unimodular matrices. *Proceedings of the American Mathematical Society*, 16, 1965.

E. Camponogara and P. H. R. Nakashima. Optimizing gas-lift production of oil wells: Piecewise linear formulation and computational analysis. *IIE Transactions*, 38, 2006a. 10 pages.

E. Camponogara and P. H. R. Nakashima. Solving a gas-lift optimization problem by dynamic programming. *Accepted to appear in European Journal of Operational Research*, 2006b.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

G. Cornuejols, M. L. Fisher, and G. L. Nemhauser. On the uncapacitated location problem. *Annals of Discrete Mathematics*, 1:163–177, 1977.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 1979.

F. Jahn, M. Cook, and M. Graham. *Hydrocarbon Exploration and Production*. Elsevier, 2003.

E. P. Kanu, J. M. Mach., and K. E. Brown. Economic approach to oil production and gas allocation in continuous gas lift. *Journal of Petroleum Technology*, pages 1887–1892, October 1981.

G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York, NY, 1988.

N. Nishikiori, R. A. Redner, D. R. Doty, and Z. Schmidt. An improved method for gas lift allocation optimization. *ASME Journal of Energy Resources Technology*, 117:87–92, 1995.

L. A. Wolsey. *Integer Programming*. John Wiley & Sons, New York, NY, 1998.