Computers & Industrial Engineering 61 (2011) 364–372

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Minimizing fuel cost in gas transmission networks by dynamic programming and adaptive discretization

Conrado Borraz-Sánchez, Dag Haugland *

Department of Informatics, University of Bergen, P.B. 7803, N-5020 Bergen, Norway

ARTICLE INFO

Article history:
Available online 22 July 2010

Keywords:
Gas transmission network
Compressor
Fuel cost
Discretization
Dynamic programming
Tree decomposition

ABSTRACT

In this paper, the problem of computing optimal transportation plans for natural gas by means of compressor stations in pipeline networks is addressed. The non-linear (non-convex) mathematical model considers two types of continuous decision variables: mass flow rate along each arc, and gas pressure level at each node. The problem arises due to the presence of costs incurred when running compressors in order to keep the gas flowing through the system. Hence, the assignment of optimal values to flow and pressure variables such that the total fuel cost is minimized turns out to be essential to the gas industry. The first contribution from the paper is a solution method based on dynamic programming applied to a discretized version of the problem. By utilizing the concept of a tree decomposition, our approach can handle transmission networks of arbitrary structure, which makes it distinguished from previously suggested methods. The second contribution is a discretization scheme that keeps the computational effort low, even in instances where the running time is sensitive to the size of the mesh. Several computational experiments demonstrate that our methods are superior to a commercially available local optimizer.

1. Introduction

Natural gas has become one of the most important energy resources worldwide. Consequently, the volumes of gas flowing from the fields through transmission networks to the market have been increasing steeply during the past decades, and in parallel, a growing interest in reducing costs associated with pipeline gas transportation has been observed.

A gas transmission network is a system consisting of sources, pipelines, compressors and distribution centers. At the sources, a supply of gas received from external fields is refined, and transported via pipelines and compressors to the distribution centers. The distribution centers are the end points of the transmission network, and the gas finally received here is input to local distribution networks supporting households and other clients.

The flow capacity of any pipeline increases with the inlet pressure and decreases with the outlet pressure of the pipeline. If no compressors are installed along a flow path, the pressure will be continuously decreasing. Since the pressure at the distribution centers typically is fixed, the flow capacity may therefore eventually become prohibitively small. To increase the pressure, and thereby the flow capacity, compressors are hence installed at the entry points of selected pipelines. Operation of the compressors incurs a cost depending on the flow and their inlet and outlet pressures.

In this paper, the fuel cost minimization problem (FCMP) to transport natural gas in a general class of transmission networks is addressed. The FCMP involves two types of continuous decision variables: mass flow rate through each arc, and gas pressure level at each node. The problem is to determine a transportation plan minimizing the total fuel cost, while meeting a specified demand at the distribution centers.

An extensive literature on the FCMP has been published over the past 30 years. Most of the suggested solution methods are limited to pipelines networks with acyclic structures, and in such instances, the suggested methods have shown a strong potential. In some of the more recent works, methods for cyclic networks have been developed. However, since these optimization approaches require a certain sparse network structure, their applicability is somewhat restricted. The following sections give a more detailed overview of the most relevant methods. A common assumption is that the system is in steady-state, which means that rapid changes in parameter values do not occur.

1.1. Methods based on dynamic programming

By discretizing the range of the pressure variables, FCMP has in several works been formulated as a combinatorial problem that can be approached by dynamic programming (DP). Wong and Larson (1968) published the first work on optimization of pipeline transportation of natural gas by DP. They applied it to a gun-barrel (linear) network, that is a problem instance where the underlying

* Corresponding author. Tel.: +47 55 58 40 33; fax: +47 55 58 41 99.
E-mail addresses: Conrado.Borraz-Sanchez@ii.uib.no (C. Borraz-Sánchez), dag@ii.uib.no (D. Haugland).

0360-8352/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2010.07.012

network is a path, using a recursive formulation. A disadvantage was that the length and diameter of the pipeline segment were assumed to be constant because of limitations of DP. Martch and McCall (1972) modified the problem by adding branches to the pipeline segments and letting the length and diameter of the pipeline segments vary. However, since their problem formulation did not allow unbranched network, more complicated network systems could not be handled.

The first attempt to solve instances with tree-shaped networks by DP was done by Zimmer (1975). A similar approach was described by Lall and Percell (1990). They allowed a divergent branch in their systems and included an integer decision variable into the model that represented the number of operating compressors in the stations.

Carter (1998) developed an algorithm referred to as non-sequential DP. The principal idea of the method is to reduce the network by three basic reductions techniques until it consists of a single node. The method can handle a wide range of instances with cyclic networks, but fails if the networks are not sufficiently sparse. Based on this approach, Borraz-Sánchez and Ríos-Mercado (2004, 2009) developed a hybrid meta-heuristic combining tabu search and non-sequential DP. The restriction that the networks must be sparse is however a shortcoming that the hybrid method inherits from the original paper.

1.2. Methods based on gradient techniques

Percell and Ryan (1987) applied a generalized reduced gradient (GRG) method for solving FCMP. In comparison with DP, an advantage of GRG is that the rapid growth in instance size caused by many discretization points is avoided. Also, GRG is applicable to cyclic networks. Nonetheless, only a local optimum can be provided, of which instances of FCMP can have many, and the solution to be output depends on the choice of starting point. Flores-Villarreal and Ríos-Mercado (2003) extended the previous study by means of an extensive computational evaluation of the GRG method.

1.3. Other techniques and related problems

Wu, Ríos-Mercado, Boyd, and Scott (2000) address the non-convex nature of FCMP, and suggest mathematical models that provide strong relaxations, and hence tight lower bounds on the minimum cost. Based on this model and the PhD thesis of Wu (1998), they demonstrated the existence of a unique solution to a non-linear algebraic equations system over a set of flow variables. This theoretical result lead to a technique for reducing the size of the original network without altering its mathematical structure.

Villalobos-Morales, Cobos-Zaleta, Flores-Villarreal, Borraz-Sánchez, and Ríos-Mercado (2003) formulated a non-linear optimization model that also contains integer variables representing the number of compressor units inside a compressor station. Cobos-Zaleta and Ríos-Mercado (2002) extended this model, and suggested a solution technique based on outer approximation.

1.4. Contributions from the current work

Several works have demonstrated that, at least in acyclic and sparse cyclic instances of FCMP, it is a promising approach to discretize the pressure variables, and apply DP to the resulting combinatorial problem. The purpose of this research is twofold: First, we demonstrate how such approaches can be applied to networks of arbitrary structure. Second, in order to keep the running time down in dense and cyclic instances, we propose a new scheme for discretizing the pressure variables. This scheme is adaptive in the sense that it avoids fine discretization of variables in area unlikely to contain good solutions, and intensifies discretization in more promising regions.

The remainder of the paper is organized as follows: In the next section, we define the problem in mathematical terms. In Section 3, we present a contemporary solution method, and point out a simple instance where it fails. In Section 4, we show how the weakness of the method discussed in Section 3 can be overcome by our alternative method. Our adaptive discretization method is given in Section 5. Results from computational experiments are reported in Section 6, and concluding remarks are given in Section 7.

2. Problem definition

Let $G = (V,A)$ be a directed graph representing a gas transmission network, where V and A are the node and arc sets, respectively. Let V_v^+ and V_v^- denote the sets of out- and in-neighbors, respectively, of node $v \in V$. Let $V_s \subseteq V$ be the set of supply nodes representing the sources, $V_d \subseteq V$ the set of demand nodes representing the distribution centers, and let $A = A_c \cup A_p$ be partitioned into a set of compressor arcs A_c and a set of pipeline arcs A_p. That is, if $(u, v) \in A_c$ then $u, v \in V$ are the network nodes representing the input and the output units, respectively, of some compressor (u, v). An analogous interpretation is made for pipeline arcs $(u, v) \in A_p$.

Two types of decision variables are defined: Let x_{uv} denote the mass flow rate at arc $(u, v) \in A$, and let p_v denote the gas pressure at node $v \in V$. For each $v \in V$, we define the parameters net mass flow rate B_v and pressure bounds P_v^L and P_v^U (lower and upper, respectively). By convention, $B_v > 0$ if $v \in V_s$, $B_v < 0$ if $v \in V_d$, and $B_v = 0$ otherwise. By the assumption that flow is conserved at the nodes, the decision variables are subject to the constraints $\sum_{u \in V_v^+} x_{vu} - \sum_{u \in V_v^-} x_{uv} = B_v$ for all $v \in V$. Constraints linking the pressure and flow variables are given for the arc sets A_c and A_p, and these are discussed next.

2.1. Compressor arc constraints

The variables that are manipulated in a compressor $(u, v) \in A_c$ in order to have the desired values of x_{uv}, p_u, and p_v are according to Wu et al. (2000) compressor speed S_{uv}, volumetric inlet flow rate Q_{uv}, adiabatic head H_{uv} and adiabatic efficiency η_{uv} of the compressor. These can briefly be explained as follows (more details can be found in the cited work):

- The variable S_{uv} is the speed at which each molecule flows through compressor (u, v), and should not be confused with the flow x_{uv} itself.
- While x_{uv} is the mass flow per time unit, the volumetric flow Q_{uv} is simply x_{uv} divided by the gas density at the inlet point of the compressor. Due to pressure variations, the density is not constant throughout the network.
- The adiabatic head H_{uv} says how much energy is required to compress one mass unit of gas from one pressure level to another without altering the gas temperature.
- The adiabatic efficiency η_{uv} is the ratio between the energy effective in compressing the gas and the total energy spent.

As explained more detailed by, e.g. Wu et al. (2000), the above magnitudes relate to (x_{uv}, p_u, p_v) according to

$$H_{uv} = \alpha\left[\left(\frac{p_v}{p_u}\right)^{\kappa} - 1\right] \quad \forall (u, v) \in A_c \tag{1}$$

$$Q_{uv} = \alpha\kappa\frac{x_{uv}}{p_u} \quad \forall (u, v) \in A_c \tag{2}$$

$$\frac{H_{uv}}{S_{uv}^2} = \phi^1\left(\frac{Q_{uv}}{S_{uv}}\right) \quad \forall (u, v) \in A_c \tag{3}$$

$$\eta_{uv} = \phi^2\left(\frac{Q_{uv}}{S_{uv}}\right) \quad \forall (u, v) \in A_c \tag{4}$$

where $\kappa \in (0,1)$ and $\alpha > 0$ are gas specific constants, and ϕ^1 and ϕ^2 are polynomial functions (typically of degree 3). The coefficients of ϕ^1 and ϕ^2 are assessed by applying least squares analysis to a set of selected data points. For each $(u,v) \in A_c$, Q_{uv} is subject to lower and upper bounds Q^L_{uv} and Q^U_{uv}, and we adopt a similar notation for bounds on the variables S_{uv},H_{uv} and η_{uv}.

The fuel consumption cost is given by Wu et al. (2000):

$$g_{uv}(x_{uv},p_u,p_v) = \frac{cx_{uv}\left[\left(\frac{p_v}{p_u}\right)^{\kappa}-1\right]}{\eta_{uv}} \quad \forall (u,v) \in A_c$$

where $c > 0$ is a monetary constant.

The feasible *operating domain* of compressor station $(u,v) \in A_c$ is the set $D_{uv} \subset \Re^3$ of value assignments to (x_{uv},p_u,p_v) for which there exist values of $(Q_{uv},S_{uv},H_{uv},\eta_{uv})$ satisfying (1)–(4) and the bounds $Q^L_{uv} \leqslant Q_{uv} \leqslant Q^U_{uv}$, $S^L_{uv} \leqslant S_{uv} \leqslant S^U_{uv}$, $H^L_{uv} \leqslant H_{uv} \leqslant H^U_{uv}$, and $\eta^L_{uv} \leqslant \eta_{uv} \leqslant \eta^U_{uv}$.

We assume that for all $(x_{uv},p_u,p_v) \in D_{uv}$, $\forall (u,v) \in A_c$, there is a *unique* feasible $(Q_{uv},S_{uv},H_{uv},\eta_{uv})$. This correspondence defines the desired transformation from feasible flow and pressure variable values (x_{uv},p_u,p_v) to an estimate $g_{uv}(x_{uv},p_u,p_v)$ of the fuel cost.

2.2. Pipeline arc constraints

Following Wu et al. (2000), the relation between pipeline flow and (sufficiently high) pressure in steady state networks can be written as $x^2_{uv} = W_{uv}(p^2_u - p^2_v)$, where $W_{uv} > 0$ is some constant depending on characteristics of the gas and the pipeline $(u,v) \in A_p$.

2.3. Mathematical model

For each node $v \in V$, we impose lower and upper pressure bounds P^L_v, and P^U_v, respectively. We confine our study to irreversible flow, and impose $x_{uv} \geqslant 0$ for all $(u,v) \in A$. Summarizing the two last sections, the FCMP can then be formulated as follows:

$$\min \quad \sum_{(u,v)\in A_c} g_{uv}(x_{uv},p_u,p_v) \tag{5}$$

$$\text{s.t.:} \quad \sum_{u\in V^+_v} x_{vu} - \sum_{u\in V^-_v} x_{uv} = B_v \quad \forall v \in V \tag{6}$$

$$(x_{uv},p_u,p_v) \in D_{uv} \quad \forall (u,v) \in A_c \tag{7}$$

$$x^2_{uv} = W_{uv}(p^2_u - p^2_v) \quad \forall (u,v) \in A_p \tag{8}$$

$$P^L_v \leqslant p_v \leqslant P^U_v \quad \forall v \in V \tag{9}$$

$$x_{uv} \geqslant 0 \quad \forall (u,v) \in A \tag{10}$$

Wu et al. (2000) give simple illustrations of the domains D_{uv}, pointing out the fact that these typically are non-convex sets. It is therefore unlikely that simple local optimization methods are sufficient to solve the above model, and the remainder of this article is devoted to methods aimed for non-convex problem instances.

3. Solution methods

Several solution methods have been suggested for FCMP, including those by Ríos-Mercado, Kim, and Boyd (2006) and Borraz-Sánchez and Ríos-Mercado (2009), which all follow the idea of Algorithm 1.

Algorithm 1. SolveFCMP

Step 1: Choose initial (feasible) flow
repeat
  Step 2: Optimize pressure while keeping the flow fixed
  Step 3: Optimize flow while keeping the pressure fixed
until flow does not change

With the risk of missing the global optimum, flow and pressure are determined separately in Steps 2 and 3, respectively. As we show next, this can be accomplished by focusing on only a subset of the variables.

3.1. Compressor network

The focus in this paper is to accomplish Step 2 of the above algorithm, and we now demonstrate how this can be done by optimizing over only a subset of the pressure variables.

Let $V' \subseteq V$ consist of exactly one node from each of the connected components in the directed graph (V,A_p), and let $G^v = (V^v,A^v)$ denote the component (subgraph) to which $v \in V'$ belongs. Define the *compressor network* (by Ríos-Mercado, Wu, Scott, & Boyd (2002) referred to as the *reduced* network) as the directed graph $G' = (V',A'_c)$, where $(u,v) \in A'_c$ if and only if $u,v \in V'$ and there exists some arc in A_c from V^u to V^v. As in (Ríos-Mercado et al., 2002), we assume that G' does not contain loops, which means that no compressor arc has both its start node and its end node in the same connected component of (V,A_p). Equivalently, the node set of G' can be associated with the subgraphs G^v $(v \in V')$, as shown in the illustration of the transition from G to G' (Fig. 1).

Theorem 1. *If $A_c = \emptyset$ then for any $B \in \Re^V$ satisfying $\sum_{v\in V} B_v = 0$, any real number $p^{ref} \geqslant 0$, and any $v \in V$, there exist unique $x \in \Re^A$ and $p \in \Re^V_+$ satisfying $p_v = p^{ref}$, (6) and (8).*

Proof. See Ríos-Mercado et al. (2002). □

The essence of Theorem 1 is that in any network consisting exclusively of pipeline arcs, the flow and pressure values are all gi-

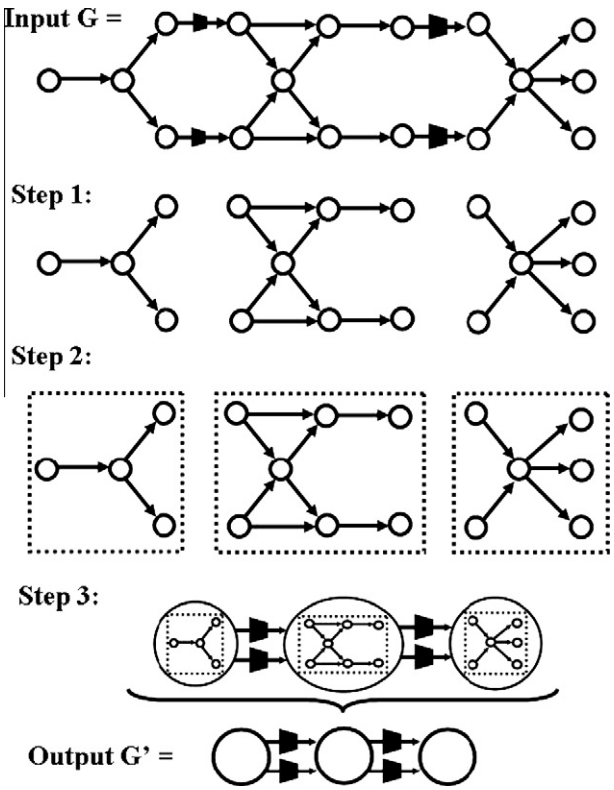

Fig. 1. Transition to compressor network.

ven uniquely once the pressure at any reference node $v \in V$ is set to any value p^{ref}. If (x,p) also satisfies (9) and (10), the assignment $p_v = p^{ref}$ is feasible.

The observation that Theorem 1 applies to G^v for all $v \in V'$ suggests the following approach to Step 2 in Algorithm 1: Identify the connected components in (V,A_p), and nominate one reference node in each. Since x is fixed in this step, all other pressure values are found by utilizing (8), and feasibility is checked by verifying whether (9) holds. As pointed out by Ríos-Mercado et al. (2006), and exploited in the algorithm given in the same reference, it follows that Step 2 is reduced to the problem of solving instances of (5)–(10) where $A_p = \emptyset$ and x is fixed.

Theorem 1 shows that if x_{uv} and p_v are fixed for all compressor arcs $(u,v) \in A_c$ and all reference nodes $v \in V'$, the remaining variable values are computed by solving the system of equations consisting of (6) and (8). In Step 3, we thus keep p_v fixed for all $v \in V'$, and optimize over $\{x_{uv}:(u,v) \in A_c\}$. To respect the flow balance constraints (6), flow updates must be made by sending flow along cycles in G', and by identifying cycles with negative net cost a reduction in the objective function value is achieved. To check the cost of sending flow along a cycle, we have to take into account the change in x_{uv} for all compressor arcs (u,v) along the cycle, but also the change in p_v for all $v \in V \setminus V'$ in connected components of (V,A_p) intersected by the cycle. For more details, we refer the reader to Ríos-Mercado et al. (2002).

Step 3 will not be discussed further here. We define the problem FCMP' to be equivalent to (5)–(10), with the additional conditions that $A_p = \emptyset$ and x is fixed. With the purpose of developing efficient computational methods for Step 2, the rest of the paper is devoted to problem FCMP'.

3.2. Discretized pressure and dynamic programming formulation

Carter (1998) suggested to solve FCMP' by discretizing $[P^L,P^U]$ and then apply a network reduction technique referred to as non-sequential dynamic programming (NDP). Assume that there are τ discretization points $p_v^1,\ldots,p_v^\tau$ for each $v \in V$ such that $P_v^L \leqslant p_v^1 < \cdots < p_v^\tau \leqslant P_v^U$, and for all $i,j = 1,\ldots,\tau$, let $g_{uv}^{ij} = g_{uv}\left(x_{uv},p_u^i,p_v^j\right)$ if $\left(x_{uv},p_u^i,p_v^j\right) \in D_{uv}$ and $g_{uv}^{ij} = \infty$, otherwise. Then NDP consists of a sequence of reductions of G until the resulting graph is a single node. Three reduction types (see Fig. 2) are considered:

(a) *Serial*: If $v \in V$ has exactly two incident arcs (u,v) and (v,t) in G, then v, (u,v) and (v,t) are replaced by a new arc (u,t), and $g_{ut}^{ij} = \min_k\left\{g_{uv}^{ik} + g_{vt}^{kj} :\ k = 1,\ldots,\tau\right\}$. The same principle applies if both arcs incident to v enter (leave) v.

(b) *Dangling*: If $v \in V$ has only one incident arc (v,t), then t and (v,t) are removed, and, for all in-neighbors u of v in G, g_{uv}^{ij} is updated to $g_{uv}^{ij} + \min_k\left\{g_{vt}^{ik} :\ k = 1,\ldots,\tau\right\}$. Similar updates apply to the out-neighbors of v, and the principle applies also if the sole neighbor of t is an out-neighbor.

(c) *Parallel*: If $k > 1$ arcs $a_1,\ldots,a_k$ in G connect nodes u and v, then these are replaced by a single arc (u,v). The associated cost parameters are defined as $g_{uv}^{ij} = \sum_{\ell=1}^k g_{a_\ell}^{ij}$ $\forall i,j = 1,\ldots,\tau$.

The serial and parallel reductions constitute the pre-processing procedure suggested by Koster, van Hoesel, and Kolen (1999).

When neither of the reductions (a)–(c) can be carried out, NDP fails. Fig. 3 shows a simple example where this occurs. To overcome this weakness, we now go on to demonstrate how such instances of FCMP' can be solved.

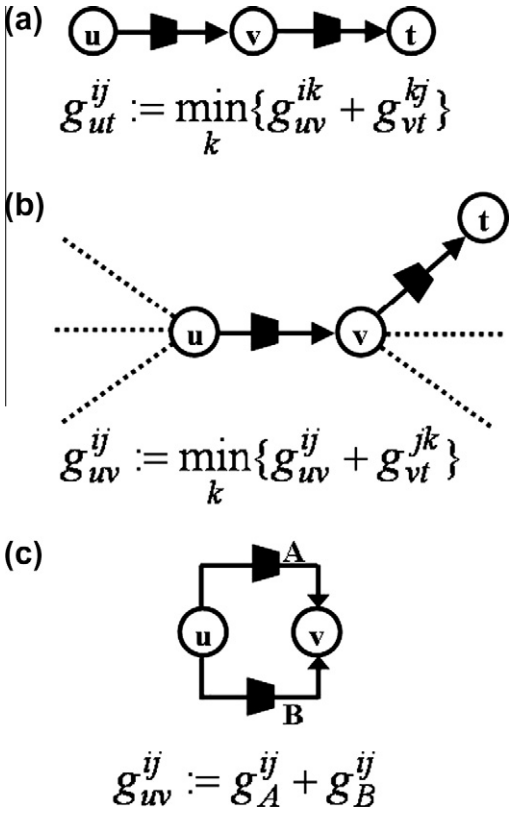

Fig. 2. Network reduction types.

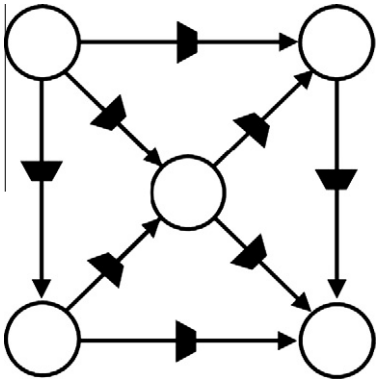

Fig. 3. An instance of G where NDP fails.

4. A tree decomposition approach to optimizing the pressure values

FCMP' has the mathematical structure of the *frequency assignment problem* (Koster et al., 1999), and can also be solved by the procedure suggested in the cited reference. This is based on the following concept introduced by Robertson and Seymour (1986):

Definition 1. A tree decomposition of G is a pair $\mathcal{J} = (\{X_i : i \in I\},T)$, where each X_i is a subset of V, called a bag, and T is a tree with node set I. The following properties must be satisfied:

- $\bigcup_{i\in I} X_i = V$;
- for all $(u,v) \in A$, there is an $i \in I$ such that $\{u,v\} \subseteq X_i$;

- $\forall i,j,k \in I$, if j lies on the path between i and k in T, then $X_i \cap X_k \subseteq X_j$.

The width of a tree decomposition $\mathscr{J}$ is $\max_{i\in I}|X_i| - 1$.

For any $X \subseteq V$, define p_X as the vector with components p_v ($v \in X$) in any consistent order. Define $\mathscr{D}_v = \{p_v^1, \ldots, p_v^\tau\}$ for all $v \in V$, and let $\mathscr{D}_X = \{p_X : p_v \in \mathscr{D}_v, \forall v \in X\}$. For any $i \in I$, let K_i denote the set of child nodes of i in T.

Algorithm 2. DP $(\mathscr{J}, i, X, \pi)$

if i is a leaf in T **then**
return

$$\min_{p\in\mathscr{D}_{X_i\cup X}} \left\{ \sum_{\substack{(u,v)\in A \\ u,v\in X_i\cup X}} g_{uv}(x_{uv}, p_u, p_v) : p_v = \pi_v \forall v \in X \right\}$$

else
return

$$\min_{p\in\mathscr{D}_{X_i\cup X}} \left\{ \sum_{\substack{(u,v)\in A \\ u,v\in X_i\cup X}} g_{uv}(x_{uv}, p_u, p_v) + \sum_{j\in K_i} \mathbf{DP}(\mathscr{J}, j, X_i \cup X, p) : p_v = \pi_v \forall v \in X \right\}$$

Algorithm 2 applies dynamic programming to a tree decomposition $\mathscr{J}$ of G. When bag X_i is processed, the union X of all ancestor bags of X_i are input along with a pressure vector $\pi \in \mathscr{D}_X$. The algorithm optimizes the value of p_v for all $v \in X_i$ by complete enumeration of $\mathscr{D}_v$, and by taking into account optimal pressure assignments to all nodes in all child bags of X_i. This is expressed in terms of a recursive call in Algorithm 2. Since $X_i \cap X$ may be non-empty, we must ensure that nodes contained in this set are not assigned new pressure values when processing X_i, and we impose the constraint that $p_v = \pi_v$ for all $v \in X$.

The running time of Algorithm 2 is $\mathcal{O}(|I|\tau^d)$, where d is the width of $\mathscr{J}$. This means that finding a tree decomposition of small width can be crucial for the running time of the algorithm. It is however well known (Robertson & Seymour, 1986) that finding one with minimum width is an NP-hard problem, and it is therefore unlikely that a tree decomposition minimizing the running time of Algorithm 2 can be found in polynomial time. We will rely on a heuristic approach to constructing $\mathscr{J}$ with small width.

4.1. Pre-processing variable bounds

In this section, we propose a bounding technique to be applied as a pre-processing technique in order to speed-up the convergence of our proposed methods. The aim of applying this pre-processing technique is to avoid as much as possible huge computational efforts when applying a finer discretization. We basically shrink all pressure bounds in G based on the maximum and minimum potential pressure values given by the physical properties in each compressor arc $(u, v) \in A$. We can define this bounding technique as elementary operations that may lead to better algorithmic properties before attempting to apply any of our proposed methods to G.

Given $\left[P_u^L, P_u^U\right], \forall u \in V$, the new refined pressure bounds can then be expressed as:

$$lb_u(P^L, P^U) = \max\left\{P_u^L, \Pi_u^L\right\} \leqslant p_u \leqslant ub_u(P^L, P^U) = \min\left\{P_u^U, \Pi_u^U\right\} \tag{11}$$

where (1) is used to obtain the pressure bounds

$$\Pi_u^L = \max_{v:(u,v)\in A} P_v^L \left(\frac{\kappa H_{uv}^U}{ZRT_s} + 1\right)^{-(1/\kappa)} \tag{12}$$

and

$$\Pi_u^U = \min_{v:(u,v)\in A} P_v^U \left(\frac{\kappa H_{uv}^L}{ZRT_s} + 1\right)^{-(1/\kappa)} \tag{13}$$

with κ as the isotropic factor defined as

$$\kappa = \frac{(1.287 - 1)}{1.287} \approx 0.223.$$

5. An adaptive discretization method

An important parameter of the approach suggested in the previous section, is the number of discretization points, τ, by which we represent each pressure variable. Assessing this parameter may be difficult. On the one hand, a large value of τ increases the possibility of finding a feasible solution of good quality. On the other hand, the previous section showed that the asymptotic increase in the running time is proportional to τ^d. With a large width d of the tree decomposition, choosing a large value of τ may lead to a slow method.

In this section, we therefore develop a method where the number of discretization points is initially small, and upgraded by a fixed factor until at least one feasible point is found by dynamic programming. Next, for each solution in a selection of the feasible ones hence found, we define an enclosing rectangular subset of the solution set, henceforth referred to as a *focus area*. The same procedure is then applied to each focus area.

By this approach, we focus the search in the neighborhood of some of the feasible solutions found, and repeat the idea recursively until the discretization distance within the focus area drops below a given threshold. The idea can be depicted by a search tree S (see Fig. 4), where each node corresponds to a unique focus area and the branches correspond to the set of feasible solutions found by DP and selected for further exploration. To limit the size of the search tree, only a fixed proportion of the feasible solutions are selected to be explored. If Ω is the set of feasible solutions found, we select the $\lceil\sigma|\Omega|\rceil$ solutions in Ω with the smallest cost, where $\sigma \in (0, 1]$ is an input parameter.

The dynamic programming algorithm (Algorithm 2) can easily be generalized such that it produces a set of solutions rather than only the best solution found. For all possible value assignments to the variables corresponding to the root bag of $\mathscr{J}$, we make optimal value assignments to all the remaining variables. Hence, $|\Omega| \leqslant \tau^{|X_0|}$, where X_0 is the root bag of $\mathscr{J}$. Only a trivial modification of Algorithm 2, where the root of $\mathscr{J}$ is treated differently from the other bags, is needed, and for reasons of brevity we omit the details. The resulting algorithm, denoted by DP', returns Ω and takes as input the same arguments as does Algorithm 2.

If DP' returns the empty set when τ is set equal to an initial number τ_0 of discretization points, we update τ by a fixed factor γ and call DP' again. The process is repeated until $\Omega \neq \emptyset$ or $\Delta_v = \frac{P_v^U - P_v^L}{\tau - 1} < \epsilon$ for all $v \in V$, where the threshold value ϵ is an input parameter.

The focus area around any selected solution $p \in \Omega$ is defined as the Cartesian product of the intervals $[lb_v(p - \Delta), ub_v(p + \Delta)]$, where $\Delta \in \Re^V$ is the vector with components $\Delta_v (v \in V)$. Hence, the range of a variable in the child node covers at most two consecutive intervals between discretization points in the parent node.

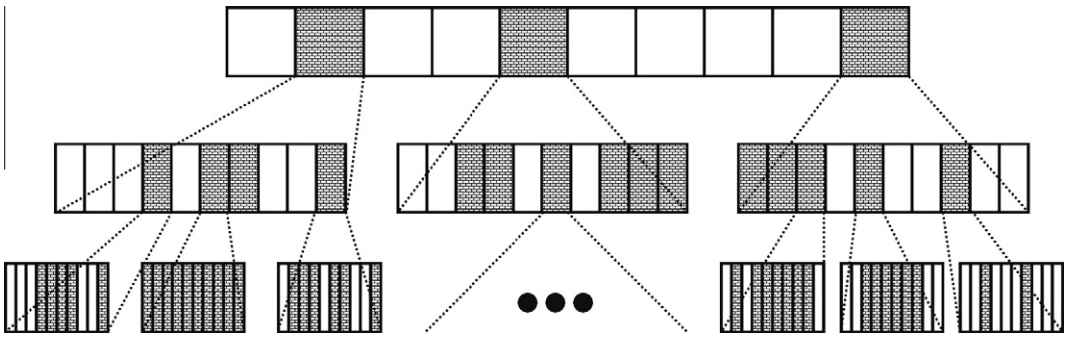

Fig. 4. Search tree based on adaptive discretization.

However, the improved bounds defined by (11) are likely to narrow down the range.

A summary of the approach is given in Algorithms 3 and 4.

Algorithm 3. adaptiveDiscretization $(\mathscr{J}, G, p^L, p^U, \epsilon, \sigma, \tau_0, \gamma)$

$(\Omega, \tau) \leftarrow$ findFeasibleSolutions$(\mathscr{J}, G, p^L, p^U, \epsilon, \sigma, \tau_0, \gamma)$
$z \leftarrow \infty$
if $\Omega \neq \emptyset$ **then**
  $z \leftarrow \min\left\{\sum_{(u,v)\in A} g_{uv}(x_{uv}, p_u, p_v) : p \in \Omega\right\}$
  Let $\Omega' \subseteq \Omega$ consist of the $\lceil \sigma|\Omega| \rceil$ solutions in Ω with smallest cost
  $\Delta \leftarrow \frac{1}{\tau - 1}(P^U - P^L)$
  for all $p \in \Omega'$ **do**
    $z = \min\{z,$ adaptiveDiscretization $(\mathscr{J}, G, lb(p^L - \Delta), ub(p^U + \Delta), \epsilon, \sigma, \tau_0, \gamma)\}$

Algorithm 4. findFeasibleSolutions$(\mathscr{J}, G, p^L, p^U, \epsilon, \sigma, \tau_0, \gamma)$

$\tau \leftarrow \tau_0$
repeat
  $\Omega \leftarrow \mathrm{DP}'(p^L, p^U, \tau)$
  $\tau \leftarrow \lceil \gamma\tau \rceil$
until $\Omega \neq \emptyset$ or $\frac{p_v^U - p_v^L}{\tau - 1} < \epsilon \ \forall v \in V$
return (Ω, τ)

6. Numerical experiments

6.1. Overview of the experiments

In the first experiment, we examine the performance of the dynamic programming approach when the number of discretization points is kept fixed. We let $\tau \in \{50, 100, 1000\}$, and let the pressure values be uniformly distributed between their lower and upper bounds. The purpose of the experiment is to study the impact of τ on the quality of the solution and the running time.

In the second experiment, we analyze the performance of the adaptive discretization approach, and compare it to fixed discretization. The idea behind the experiment is to investigate whether adaptive discretization produces solutions comparable to those of fixed discretization in less computer time.

The third experiment is a similar comparison between the dynamic programming approaches and the commercially available local optimizer, *MINOS* (Murtaugh & Saunders, 1983). Since the local optimum output by MINOS, if any, turns out to be sensitive to the starting point, we run MINOS for 500 and 1000 randomly generated starting points. The points are drawn from the uniform distribution on $[p^L, p^U]$.

The fourth experiment is a comparison between the solutions produced by our methods to (a lower bound on) the true minimum cost. We submit FCMP' to the generic global optimization tool, *BARON* (Tawarmalani & Sahinidis, 2004), which is an implementation of a variant of branch-and-bound where a convex program is solved in each node of the search tree. To solve the convex subproblems, BARON is set to call MINOS. We impose a time limit of 3600 CPU-seconds on each application of BARON, and the relative optimality tolerance be 0.01. That is, any feasible solution is considered to be optimal if the gap between the objective function value and its lower bound is below one percent of the objective function value. In instances where BARON fails to compute the global optimum, it may still provide a lower bound on the minimum cost, and this bound may also give some indications on the quality of the output from our methods.

Our solution procedures were coded in C++ under Linux Red Hat, and all experiments were run on a 2.4 GHz Intel(R) processor with 2 GByte RAM. To compute the tree decomposition $\mathscr{J}$ to be input to the dynamic programming algorithms, we apply the technique given by Subbarayan (2007) based on *Maximum Cardinality Search* (Tarjan & Yannakakis, 1984). Experiments with BARON and MINOS were conducted by formulating the model in GAMS (GAMS Development Corporation, 2008), and we have used version 8.1.5 of BARON and version 5.51 of MINOS.

6.2. Test instances

All experiments reported in this work were carried out on the set of test instances shown in Table 1. Each row gives an identifier of an instance, the size in terms of nodes and arcs in G after reduction, and the type of compressor used. We consider nine different compressor types, and all compressors are identical within any given instance. Furthermore, the width and the number of bags in the tree decomposition are given in the two last columns of Table 1.

All test instances can be downloaded in GAMS-format at http://www.ii.uib.no/~conrado/caie/instances/index.html.

6.3. Results

Table 2 shows the results achieved by fixed discretization for three different values of τ. Instance references are given in the first column, and computation times (CPU-seconds) and objective function values for the respective values of τ are given in columns 2–7. The only case where the method failed to find a feasible solution was for $\tau = 50$ in instance K. We observe that as τ increases, better solutions are found (minimum cost decreases) in all instances, except from a cost increase from $\tau = 50$ to $\tau = 100$ in instances O and P. Nevertheless, a finer discretization also implies, as expected, that

Table 1
Test instances.

Ref	Size		Type	$\mathscr{J}$	
	$\lvert V'\rvert$	$\lvert A'_c\rvert$		Width	$\lvert I\rvert$
A	3	3	3	3	1
B	3	3	4	3	1
B1	3	3	5	3	1
B2	3	3	6	3	1
B3	3	3	8	3	1
C	4	6	1	3	3
D	4	6	2	3	4
E	4	6	3	3	5
F	4	6	4	3	4
G	4	6	5	3	4
H	4	6	6	3	3
I	4	6	7	3	4
J	5	8	4	3	6
K	5	8	8	3	4
L	9	20	4	4	9
M	9	20	5	4	9
N	18	25	2	4	15
O	18	25	4	4	15
P	18	25	9	3	18
Q	8	10	4	3	6
R	8	10	6	3	6
S	17	23	6	4	8

Table 2
Performance of dynamic programming with fixed discretization.

Ref	$\tau = 50$		$\tau = 100$		$\tau = 1000$	
	CPU (secs)	Obj ($\times 10^6$)	CPU (secs)	Obj ($\times 10^6$)	CPU (secs)	Obj ($\times 10^6$)
A	0	1.12	0	0.77	1	0.75
B	0	2.63	0	2.62	2	2.62
B1	0	2.83	0	2.63	138	2.61
B2	0	3.30	0	2.98	138	2.84
B3	0	2.17	0	2.08	132	1.83
C	1	10.29	16	9.34	1935	8.93
D	0	7.45	11	7.34	1047	7.34
E	1	9.66	22	6.36	1082	5.29
F	2	6.87	30	5.69	1845	4.12
G	1	9.43	10	6.30	817	6.30
H	1	6.34	13	5.93	692	5.09
I	1	2.83	10	2.82	529	2.77
J	1	6.07	13	5.59	842	5.27
K	1	–	9	35.67	772	35.67
L	3	68.89	50	61.83	2987	61.73
M	3	89.68	40	74.80	2715	60.74
N	2	60.71	34	52.46	2554	46.00
O	6	63.03	180	63.38	3422	38.80
P	1	35.25	23	37.67	2417	26.54
Q	3	23.31	80	21.32	3015	15.15
R	5	24.02	149	22.78	3310	20.10
S	15	72.01	482	69.59	3662	65.51

Table 3
Performance of dynamic programming with adaptive discretization.

Ref	CPU (secs)	Calls to DP'	Obj ($\times 10^6$)
A	1	395	0.75
B	1	117	2.62
B1	0	267	2.60
B2	0	12	2.83
B3	0	11	1.82
C	1	429	7.79
D	2	16	7.35
E	1	263	5.29
F	11	852	3.94
G	1	239	5.87
H	4	319	5.17
I	4	558	2.76
J	2	122	5.18
K	2	204	31.32
L	35	210	63.14
M	16	212	54.64
N	26	631	38.09
O	31	596	29.55
P	44	547	24.34
Q	13	438	14.58
R	17	1631	15.96
S	60	815	62.46

Table 4
Performance of MINOS.

Ref	500 Starting points			1000 Starting points			*RI* (%)
	CPU (secs)	Feas (%)	Obj ($\times 10^6$)	CPU (secs)	Feas (%)	Obj ($\times 10^6$)	
A	30	59.2	0.75	64	59.4	0.75	0.0
B	59	100.0	2.63	115	99.8	2.62	0.4
B1	21	37.4	2.62	55	37.7	2.62	0.0
B2	26	22.2	2.83	57	24.3	2.83	0.0
B3	18	20.4	1.82	52	22.3	1.82	0.0
C	29	7.4	9.09	66	7.8	9.03	0.7
D	31	19.6	7.36	88	20.4	7.36	0.0
E	36	31.2	6.02	68	29.0	6.01	0.2
F	53	23.8	4.21	117	23.2	4.15	1.4
G	20	0.0	–	61	0.0	–	–
H	54	39.2	5.45	129	40.7	5.45	0.0
I	52	0.0	–	124	0.0	–	–
J	49	50.0	6.14	110	49.3	5.98	2.6
K	65	0.0	–	120	0.0	–	–
L	125	2.8	68.09	241	2.5	68.09	0.0
M	19	0.0	–	36	0.0	–	–
N	58	0.0	–	124	0.0	–	–
O	52	1.0	39.22	139	0.7	39.22	0.0
P	37	0.0	–	75	0.0	–	–
Q	147	0.0	–	365	0.0	–	–
R	102	0.0	–	238	0.0	–	–
S	57	0.0	–	116	0.0	–	–

the computational requirements increase, and the running time slightly exceeds one CPU-hour in one instance (S).

Table 3 shows the results achieved by dynamic programming and adaptive discretization. In these runs, we have used the parameter values (see Section 5) $\tau_0 = 3$, $\gamma = 1.5$, $\sigma = 0.05$ and $\epsilon = 0.001$. Instance references are given in the first column, and computation time (CPU-seconds), number of calls to DP' in Algorithm 4, and objective function values for the corresponding test instance are given in columns 2–4. We observe that the running time slightly exceeds 30 CPU-seconds in the most time-consuming instance (S).

Table 4 shows corresponding results from MINOS. Columns 2–4 give respectively the CPU-time, percentage of the 500 starting points by which MINOS found a feasible solution, and the cost of the best feasible solution found. Columns 5–8 give corresponding results for 1000 starting points. We observe that MINOS fails to find a feasible solution in 9 instances, and in the other instances (except A and B), it does so for at least 50% of the starting points. On the other hand, the solver is fast, and a relatively large number of starting points is affordable.

Tables 5 and 6 give a comparison between MINOS (with 1000 starting points) and dynamic programming with the two discretization techniques. In Table 5, we summarize and compare running times, while costs are compared in Table 6.

First, we observe from Table 5 that adaptive discretization is much faster than fixed discretization, although the latter requires only one call to the DP-algorithm. The larger number of calls to DP' reported in Table 3, seems to be more than compensated for

Table 5
Dynamic programming vs. MINOS: CPU-time.

Ref	MINOS		Discretization	
	500 Iters	1000 Iters	Fixed[a]	Adaptive
A	30	64	1	1
B	59	115	2	1
B1	21	55	138	0
B2	26	57	138	0
B3	18	52	132	0
C	29	66	1935	1
D	31	88	1047	2
E	36	68	1082	1
F	53	117	1845	11
G	20	61	817	1
H	54	129	692	4
I	52	124	529	4
J	49	110	842	2
K	65	120	772	2
L	125	241	2987	35
M	19	36	2715	16
N	58	124	2554	26
O	52	139	3422	31
P	37	75	2417	44
Q	147	365	3015	13
R	102	238	3310	17
S	57	116	3662	60

[a] With $\tau = 1000$.

Table 6
Dynamic programming vs. MINOS: cost.

Ref	Cost ($\times 10^6$)			Adaptive vs. (*RI*%)	
	MINOS[a]	Fixed[b]	Adaptive	MINOS[a]	Fixed[b]
A	0.75	0.75	0.75	0.0	0.0
B	2.62	2.62	2.62	0.0	0.0
B1	2.62	2.61	2.60	0.8	0.4
B2	2.83	2.84	2.83	0.0	0.4
B3	1.82	1.83	1.82	0.0	0.5
C	9.03	8.93	7.79	13.7	12.8
D	7.36	7.34	7.35	0.1	-0.1
E	6.01	5.29	5.29	12.0	0.0
F	4.15	4.12	3.94	5.1	4.4
G	–	6.30	5.87	–	6.8
H	5.45	5.09	5.17	5.1	-1.6
I	–	2.77	2.76	–	0.4
J	5.98	5.27	5.18	13.4	1.7
K	–	35.67	31.32	–	12.2
L	68.09	61.73	63.14	7.3	-2.3
M	–	60.74	54.64	–	10.0
N	–	46.00	38.09	–	17.2
O	39.22	38.80	29.55	24.7	23.8
P	–	26.54	24.34	–	8.3
Q	–	15.15	14.48	–	4.4
R	–	20.10	15.96	–	20.6
S	–	65.51	62.46	–	4.7

[a] With 1000 starting points.
[b] With $\tau = 1000$.

by the smaller number of discretization points. Second, Table 6 shows that in all instances but D, H and L, where fixed discretization gives up to 2.3% lower cost, the faster approach gives solutions of equal or better quality.

Also when compared to MINOS, the adaptive discretization approach turns out to be superior. Applying MINOS with a single random starting point is certainly faster, but this involves a considerable risk of failing to find a feasible solution. When the number of random starting points is increased such that the total running time of MINOS exceeds the one of adaptive discretization, the total cost of the best MINOS solution is in general higher than the cost of the solution produced by its competitor. The relative improvement of adaptive discretization when compared to MINOS and fixed discretization, respectively, is given in columns 5–6 of Table 6.

Table 7
Adaptive discretization vs. a global optimizer.

Ref	Performance of BARON				Adaptive discretization		
	#Its	#Nodes	Obj	LB	Obj	RI(%)	GAP(%)
A	551	131	0.75	0.75	0.75	0.0	0.0
B	1148	342	2.62	2.62	2.62	0.0	0.0
B1	47	5	2.62	2.59	2.60	0.4	0.8
B2	125	11	2.83	2.81	2.83	0.0	0.0
B3	37	5	1.82	1.80	1.82	0.0	0.0
C	21521	7462	9.02	4.45	7.79	13.6	42.9
D	445	38	7.35	7.28	7.35	0.0	1.0
E	17059	7023	5.30	4.02	5.29	0.2	24.0
F	26765	7480	3.94	2.71	3.94	0.0	31.2
G	5231	1283	–	2.27	5.87	–	61.3
H	2109	204	5.19	5.04	5.17	0.4	2.5
I	3267	324	–	2.73	2.76	–	1.1
J	27832	2299	5.15	5.10	5.18	−0.6	1.5
K	14968	3344	–	20.86	31.32	–	33.4
L	740	451	65.94	43.81	63.14	4.2	30.6
M	2438	765	–	31.12	54.64	–	43.0
N	1830	839	–	34.28	38.09	–	10.0
O	234	59	–	22.13	29.55	–	25.1
P	978	655	–	17.43	24.34	–	28.4
Q	330	212	17.43	12.82	14.58	16.9	11.5
R	1182	468	15.94	13.09	15.96	−0.1	18.0
S	3123	632	59.39	44.17	62.46	−5.2	29.3

Table 7 shows the performance of BARON when applied to the test instances. In addition, we compare the adaptive discretization approach to the best results obtained by BARON in terms of the quality of the solution. Columns 2–5 contain the number of iterations in BARON (the number of convex subproblems solved), the maximum number of open nodes the search tree ever had, the objective function value of the best feasible solution found (if any), and the lower bound on the minimum cost. For more convenient comparison, we give the cost obtained by adaptive discretization in column 6 (identical to column 3 in Table 3), and in the 7th column, we give whenever applicable the relative improvement (in percentages) of these solutions over the best BARON solutions.

As seen in Table 7, in 15 out of 22 instances, BARON was able to find a feasible solution, and in two instances (A and B), it was able to prove optimality within the given tolerance. In the remaining instances, no feasible solution was found before the time limit expired. By comparing columns 5 and 6, we also observe that the relative optimality gap (relative distance from minimum cost) of adaptive discretization in one instance (G) may be as large as 61.3%. In the instances where BARON found a feasible solution, the largest gap is 42.9% (instance C).

Column 7 of Table 7 shows that BARON is able to find a better solution than does our method in instances J, R and S. However, extensive computations were needed to find these solutions. The last column of the table gives the relative distance from the lower bound on optimality provided by BARON. In nine of the instances, we are less then 3% from the minimum cost, but in some instances, the optimality gap is large (as large as 61.3% in instance G). It is however unknown whether this is due to weak lower bounds or shortcomings of our algorithm.

7. Concluding remarks

In this paper, we have studied a model (FCMP) for minimizing compressor fuel cost in transmission networks for natural gas. An arc in the network model corresponds to either a pipe or a compressor, and the decision variables are arc flow and node pressure.

In addition to flow conservation constraints, the model contains non-linear constraints relating pipeline flow to inlet and outlet pressure, as well as non-convex constraints defining the operation domain of the compressors.

Following a general algorithmic idea, which has been suggested and supported experimentally in several recent works, we consider a procedure where each iteration consists of a flow improvement step and a pressure optimization step. Alternating between flow and pressure, one set of decision variables is kept fixed in each step. Still in agreement with previously suggested methods, the non-convex subproblem of optimizing pressure is approximated by a combinatorial one. This is accomplished by discretization of the pressure variables.

The contribution of this paper is a method for solving the discrete version of the problem in instances where previously suggested methods fail. Unlike methods based on successive network reductions, our method does not make any assumptions concerning the sparsity of the network. By constructing a tree decomposition of the network, and apply dynamic programming to it, we are able to solve the discrete version of the pressure optimization problem without enumerating the whole solution space. By an adaptive discretization scheme, we obtain significant speed-up of the dynamic programming approach in comparison with fixed discretization.

We have tested our solution methods on a set of imaginary instances, and compared the results to those obtained by applying both a global and a local optimizer to the continuous version of the problem. The experiments indicate that a method guaranteeing the global optimum in reasonable time seems unrealistic even for small instances. Further, discretizing the pressure variables and applying dynamic programming to a tree decomposition gives better results than applying a commercially available local optimization package.

Non-convex continuous optimization problems can in general be approached by discretization of the variable space, and in many cases, the resulting discrete problem can be solved by dynamic programming. The challenge of finding the ideal balance between accuracy in the discrete model and speed of the DP-algorithm is however hardly avoidable by the approach. We therefore believe that the adaptive discretization scheme developed in this paper may have merit beyond the specific application in gas transmission networks studied here.

References

Borraz-Sánchez, C., & Ríos-Mercado, R. Z. (2009). Improving the operation of pipeline systems on cyclic structures by tabu search. *Computers & Chemical Engineering, 33*(1), 58–64.

Borraz-Sánchez, C., & Ríos-Mercado, R. Z. (2004). A non-sequential dynamic programming approach for natural gas network optimization. *WSEAS Transactions on Systems, 3*(4), 1384–1389.

Carter, R. G. (1998). Pipeline optimization: Dynamic programming after 30 years. In *Proceedings of the PSIG meeting, Denver, CO, USA, October 1998* (pp. 1–19).

Cobos-Zaleta, D., & Ríos-Mercado, R. Z. (2002). A MINLP model for minimizing fuel consumption on natural gas pipeline networks. In *Proceedings of the XI-Latin-Ibero-American conference on operations research, Concepcion, Chile* (pp. 1–9).

Flores-Villarreal, H. J., & Ríos-Mercado, R. Z. (2003). Efficient operation of natural gas pipeline networks: Computational finding of high quality solutions. In *Proceedings of the international applied business research conference, paper 352, Acapulco, Mexico, March 2003* (pp. 1–7).

GAMS Development Corporation (2008). GAMS: The solver manuals. Washington, DC, USA.

Koster, A. M. C. A., van Hoesel, S. P. M., & Kolen, A. W. J. (1999). Optimal solutions for frequency assignment problems via tree decomposition. *Lecture Notes in Computer Science, 1665*, 338–350.

Lall, H. S., & Percell, P. B. (1990). A dynamic programming based gas pipeline optimizer. In A. Bensoussan & J. L. Lions (Eds.), *Analysis and optimization of systems* (pp. 123–132). Berlin, Germany: Springer.

Martch, H. B., & McCall, N. J. (1972). Optimization of the design and operation of natural gas pipeline systems. Paper no. SPE 4006, Society of Petroleum Engineers of AIME.

Murtaugh, B. A., & Saunders, M. A. (1983). MINOS 5.1 user's guide, technical report SOL-83-20R. Stanford, CA, USA: Systems Optimization Laboratory, Stanford University.

Percell, P. B., & Ryan, M. J. (1987). Steady-state optimization of gas pipeline network operation. In *Proceedings of the 19th PSIG annual meeting, Tulsa, USA*.

Ríos-Mercado, R. Z., Kim, S., & Boyd, E. A. (2006). Efficient operation of natural gas transmission systems: A network-based heuristic for cyclic structures. *Computers & Operations Research, 33*(8), 2323–2351.

Ríos-Mercado, R. Z., Wu, S., Scott, L. R., & Boyd, E. A. (2002). A reduction technique for natural gas transmission network optimization problems. *Annals of Operations Research, 117*(1–4), 217–234.

Robertson, N., & Seymour, P. D. (1986). Graph minors II. Algorithmic aspects of treewidth. *Journal of Algorithms, 7*, 309–322.

Subbarayan, S. (2007). An empirical comparison of CSP decomposition methods. In Hnich, B., & Stergiou, K. (Eds.), *Proceedings of the CP 2007 doctoral programme, Providence, RI, USA, September 23–27* (pp. 163–168).

Tarjan, R. E., & Yannakakis, M. (1984). Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal of Computing, 13*, 566–579.

Tawarmalani, M., & Sahinidis, N. V. (2004). Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming, 99*(3), 563–591.

Wong, P. J., & Larson, R. E. (1968). Optimization of natural gas pipeline systems via dynamic programming. *IEEE Transactions on Automatic Control, 13*(5), 475–481.

Wu, S. (1998). Steady-state simulation and fuel cost minimization of gas pipeline networks. Ph.D. Thesis. Houston, TX: University of Houston.

Wu, S., Ríos-Mercado, R. Z., Boyd, E. A., & Scott, L. R. (2000). Model relaxations for the fuel cost minimization of steady-state gas pipeline networks. *Mathematical and Computer Modeling, 31*(2-3), 197–220.

Villalobos-Morales, Y., Cobos-Zaleta, D., Flores-Villarreal, H. J., Borraz-Sánchez, C., & Ríos-Mercado, R. Z. (2003). On NLP and MINLP formulations and preprocessing for fuel cost minimization of natural gas transmission networks. In *Proceedings of the 2003 NSF design, service and manufacturing grantees and research conference, Birmingham, USA*.

Zimmer, H. I. (1975). Calculating optimum pipeline operations. Technical report, Natural Gas Company El Paso, presented at the AGA transmission meeting, Texas, USA.