A Non-Sequential Dynamic Programming Approach for Natural Gas Network Optimization

CONRADO BORRAZ-SÁNCHEZ
ROGER Z. RÍOS-MERCADO

Graduate Program in Systems Engineering
Universidad Autónoma de Nuevo León
AP 111-F, Cd. Universitaria
San Nicolás de los Garza, NL 66450
MEXICO
{conrado, roger}@yalma.fime.uanl.mx http://yalma.fime.uanl.mx/~pisis/

Abstract: The fuel cost minimization problem on steady-state natural gas pipeline networks system is addressed. From the optimization perspective, this problem is modeled as a (non-convex) nonlinear program (NLP), where we consider two types of decision continuous variables: mass flow rate through each arc and pressure value at each node. The proposed method consists of two phases. In phase 1, a set of feasible flows is found by a reduction technique, which makes use of a pre-processing procedure. Then, in phase 2, an optimal set of pressures is found (for the given flow) by applying a non-sequential dynamic programming technique. This method avoids the many numerical difficulties inherent to this very complex while treated with classical nonlinear programming techniques. We work with several different types of topologies, many of those being cyclic structures. A computational study reveals the effectiveness of the proposed procedure when tested over a wide variety of problem instances.

Key-words: Operations research, natural gas, pipeline networks, nonlinear programming, non-sequential dynamic programming.

1 Introduction

A natural gas transmission network optimization problem is addressed. It is a well-know fact, from the practical perspective, the tremendous economic impact that even a marginal improvement in network operation can have. Hence, the problem of finding out how to optimally operate the compressors driving the gas in a pipeline network becomes significantly important.

From the optimization perspective, this problem is modeled as a (non-convex) nonlinear program (NLP), where we consider two types of continuous decision variables: mass flow rate through each arc and pressure value at each node.

The state of the art on research about this problem reveals a few important facts. First, there are two fundamental types of network topologies: non-cyclic and cyclic. The former is a type that has received most of the attention during the past 30 years. Several solutions methodologies have been developed; most of them based on dynamic programming (see [6] for a survey). In contrast, cyclic topologies are a lot harder to solve. Work on this area is practically nonexistent.

In this work, we present an efficient procedure for handling cyclic structures. The procedure consists of two phases. First a set of feasible flows is found by a reduction technique and then an optimal set of pressure values (for the pre-specified flow) is found by applying a non-sequential dynamic programming (NDP) algorithm. This is motivated by the work of Carter [2]. The algorithm delivers global optimal solutions (within a given domain discretization size). This procedure avoids the many numerical difficulties inherent to this very complex problem when treated with classical nonlinear programming (NLP) techniques. Preliminary computational experience including both non-cyclic and cyclic topologies is presented. The results show the effectiveness of the proposed procedure.

The rest of the paper is organized as follows. In Section 2, we introduce and describe the NLP model. In Section 3, we present a summary of related work. The description of the algorithm is presented in Section 4. We conclude with the computational evaluation both on non-cyclic and cyclic networks, and conclusions in Section 5 and 6, respectively.

2 Mathematical Framework

2.1 Assumptions

In the present paper, we make the following modeling assumptions.

- We assume that the problem is in steady state. This is, our model will provide solution for systems that have been operating for a relative large amount of time. Transient analysis would require increasing the number of variables and the complexity of this problem, and is a fact one of the biggest research challenges in this area.
- The network is balanced. This means that the sum of all the net flows in each node of the network is equal to zero. In other words, the total supply flow is driven completely to the total demand flow without loss. Each arc in the network has a pre-specified direction.
- Each parameter is known (i.e., a deterministic model is assumed).

2.2 The NLP Model

Parameters:

V: Set of all nodes in the network
V_s: Set of supply nodes ($V_s \subseteq V$)
V_d: Set of demand nodes ($V_d \subseteq V$)
A_p: Set of pipeline arcs
A_c: Set of compressor station arcs
A: Set of all arcs in the network; $A = A_p \cup A_c$
U_{ij}: Arc capacity of pipeline (i,j); $(i,j) \in A_p$
R_{ij}: Resistance of pipeline (i,j); $(i,j) \in A_p$
P_i^L, P_i^U: Lower and upper node pressure limits; $i \in V$
B_i: Net mass flow rate at node i; $i \in N$.
$B_i > 0$ if $i \in V_s$, $B_i < 0$ if $i \in V_d$, $B_i = 0$ otherwise

Variables: x_{ij}: Mass flow rate in arc (i,j);); $(i,j) \in A$
p_i: Pressure at node i; $i \in V$

Formulation:

Minimize $\sum_{(i,j) \in Ac} g_{(i,j)}(x_{ij}, p_i, p_j)$ (1)

Subject to

$\sum_{\{j \mid (i,j) \in A\}} x_{ij} - \sum_{\{j \mid (j,i) \in A\}} x_{ji} = B_i \quad i \in V$ (2)
$x_{ij} \le U_{ij} \quad (i,j) \in A_p$ (3)
$p_i^2 - p_j^2 = R_{ij} x_{ij}^2 \quad (i,j) \in A_p$ (4)
$p_i^L \le p_i \le p_i^U \quad i \in V$ (5)
$(x_{ij}, p_i, p_j) \in D_{ij} \quad (i,j) \in A_c$ (6)
$x_{ij}, p_i \ge 0$ (7)

The objective function (1) is the sum of the fuel consumption at each compressor station in the network. Constraints (2)-(3) are the typical network flow constraints representing node mass balance and arc capacity, respectively, where $\sum_{i \in V} B_i = 0$. Constraint (4) represents the gas flow dynamics in each pipeline of the network assuming steady state. Constraints (5) denote the pressure limits in each node. Constraint (6) represents the non-convex feasible operating domain for compressor station (i,j). More details on the nature of $g_{(i,j)}$ and D_{ij} can be found in Wu et al. [10].

3 Literature Review

Dynamic Programming (DP) was invented by Richard Bellman [1] in 1957. DP for network optimization was originally applied to gun-barrel systems since the late 1960s. It was one of the most useful techniques due both to its quick computational behavior and its insensitivity to non-linearity on sequential systems.

DP was first applied to gas pipeline optimization by Wong and Larson [9] in 1968. They applied the method to fuel cost minimization in a single, straight-line system, and used a recursive formulation.

The first attempt at optimizing a branching structure in the pipeline industry using DP was by Zimmer [11] in 1975. A similar approach was described by Lall and Percell [5] in 1990, who allowed one diverging branch in their system.

In the late 1980s, hybrid DP enumeration annealing methods for optimizing more general branched and looped networks were proposed. Although these were very successful at optimizing pipelines the hybrid nature of the methods sometimes caused long run times or reduced accuracy in solving the discretized problem.

In 1989, Gilmour, Luongo, and Schroeder [4] published a hierarchical approach that allowed both loops and branches of arbitrary complexity. This was a great advance in terms of finally addressing the issue of real-world pipeline configurations. The only disadvantage was that their technique was no longer pure DP. Basically, DP was used to optimally describe the pieces of the pipeline that were arranged in a sequential manner. This typically reduced a system to a much smaller combinatory problem, but one without any possibility of a recursive DP solution. If this reduced problem was sufficiently small, it was solved exactly via enumeration; otherwise it was

solved inexactly using simulated annealing. This hierarchical approach works very well for many complex pipelines, but for others the computational cost can be very high.

Up until the early 1990s, dynamic programming could only optimize non-cyclic systems, so it has been of limited use for pipeline companies with such diverse systems. Often such companies would only consider small subsets of their pipelines when performing optimization. After such studies were performed, they could sometimes patch together the results for the different subsets manually.

The hierarchical approach depends on using specified, known flow values throughout the system. However, if the inlet and outlet flows are allowed to vary rather than being pre-specified by the user, or if internal flow splits are variable, one can apply another optimization algorithm before DP in the hierarchy to search for an optimum over these variables as well.

4 Description of Algorithm

Consider a steady-state natural gas transmission network with N compressor stations and a set of feasible flow rates. In general, there will be upper and lower limits on the pressure settings (decision variables). Rather than considering any possible pressure between the upper and lower limits, in this paper we will consider only a discrete set of k possible pressures. For instance, if our allowable pressures are between 600 and 800 PSIG, and $\Delta p=10$, we would consider only pressures at ten pound increments: 600, 610, 620, ..., 800.

Each compressor station will have a range of attainable operating conditions based on such limitations as max horsepower constraints, physical limits on individual compressor, and so on. If the compressor station can operate at specified inlet and outlet pressures, we assume that we can compute a cost for this operational setting, and that the total cost of operating the system is the sum of the cost of operating compressor station m, i_m is the integer decision variable at the suction side of the station, and j_m is the integer decision variable at the discharge side of the station.

In this part, we present a search algorithm (Fig. 1) for finding an optimal solution for the fuel cost minimization problem being addressed. This algorithm consists of two phases. The first procedure (Phase 1) makes use of a preprocessing technique to find a set feasible flow on the net. Our procedure for constructing the feasible flows, utilizes a reduction technique. Details can be found in [7].

SEARCH ALGORITHM
begin
INPUT: {Properties of natural gas pipeline network system}
G=(V,A), where A={Ad, Ac}.
B(i) sources value ∀i∈V
PHASE 1: {Find feasible flows}
Reduced_Network_PROCEDURE-1
PHASE 2: {Find optimal pressures}
Non_Sequential_DP_PROCEDURE-2
end

Figure 1. Algorithm for global optimal solutions (for fixed flows).

The second procedure (Phase 2) makes use a *Non-sequential DP Algorithm* to find optimal pressures at each node on the network (as originally proposed by Carter [2]). Rather than attempting to formulate DP as a recursive algorithm, in this approach we simply look at a system, grab two connected compressor, and replace them by a "virtual" composite element that behaves just like its components operating in an optimal manner. These elements can be selected from anywhere in the system, so the idea of "recursion" is really not a good description for this process. The process continues, reducing the number of elements in the problem by one each time, until the system is reduced no further. Typically, that occurs when there is exactly one virtual element left, which completely characterizes the optimal behavior of the entire pipeline network. The best pipeline operation can then be found by just searching one simple table for the lowest occurring value.

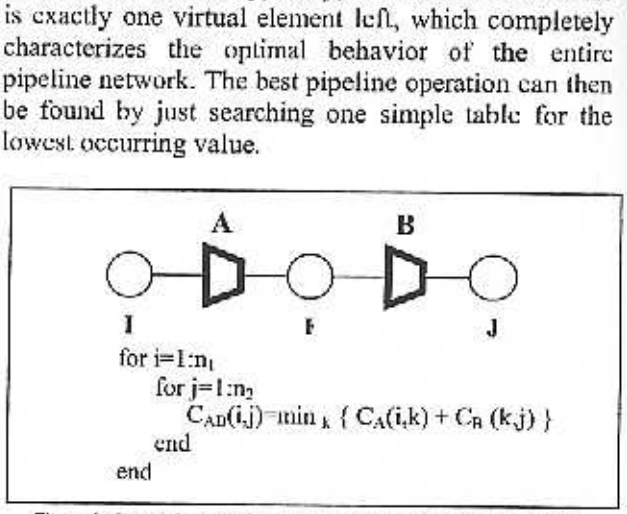

Figure 2. Combining two sequential compressor stations into a one optimal composite.

Only three types of simple composition operations

are necessary to reduce a system. Let C_A be a table costs for operating a compressor station A for its various inputs and outputs, and let C_B be similarly defined for compressor station B.

Fig. 2 shows how to combine two sequential elements A and B into one optimal composite. Here element A goes from segment I to segment K, element B goes from segment K to segment J, and the composite element goes from segment I to segment J. The composite element then has its own cost table C_{AB}. If only this transformation is allowed, the resulting method is essentially the same as the hierarchical method of Luongo [4].

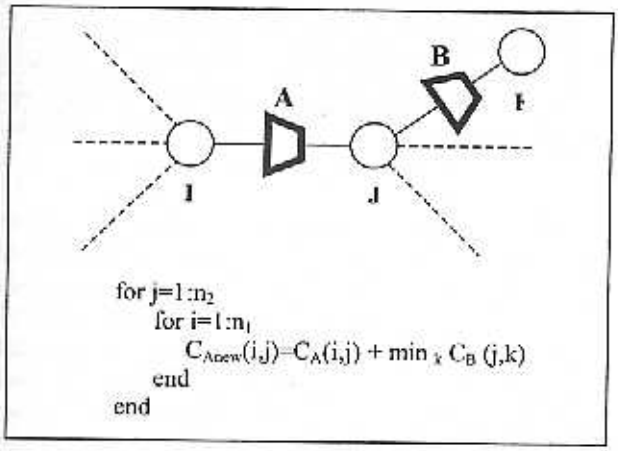

Figure 3. Combining a "dangling" element B into an adjacent element A.

Fig. 3 shows how to combine a "dangling" element B into an adjacent element A. Here element A goes from segment I to segment J, element B goes from segment J to segment K, and the composite element goes from segment I to segment J. Segment K can not be attached to any element other than B; hence the terminology "dangling element".

Fig. 4 shows how to combine two parallel compressor stations A and B into one optimal composite. Here element A goes from segment I to segment J, element B goes from segment I to segment J, and the composite element goes from segment I to segment J.

These operations can be applied to a complex network to eventually reduce it to a single composite equivalent. Note that some of the composition operations will, of course, be recursively applied to composite elements. Also, an appropriate data structure must be used to allow the reconstruction of the actual optimal pressure settings of the original system once the optimal objective has been read off the final composite table.

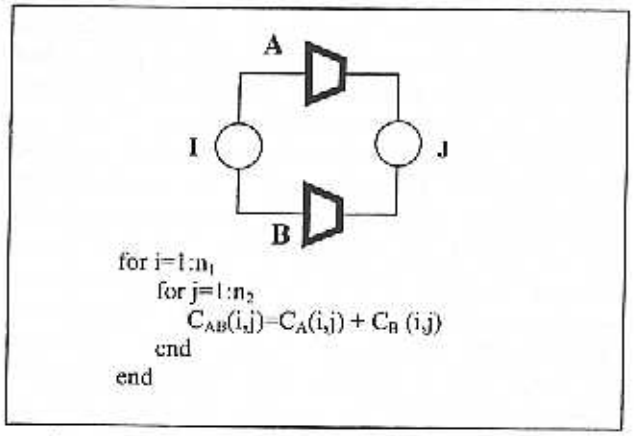

Figure 4. Combining two parallel compressor stations A and B into one optimal composite.

5 Computational Results

In order to assess the effectiveness of the proposed procedure, we apply the search NDP algorithm under different scenarios with different kinds of topologies. There are many types of topologies: (a) simple or gun-barrel, (b) tree, and (c) cyclic. Our evaluation is based on a database developed by Villalobos-Morales et al. [8].

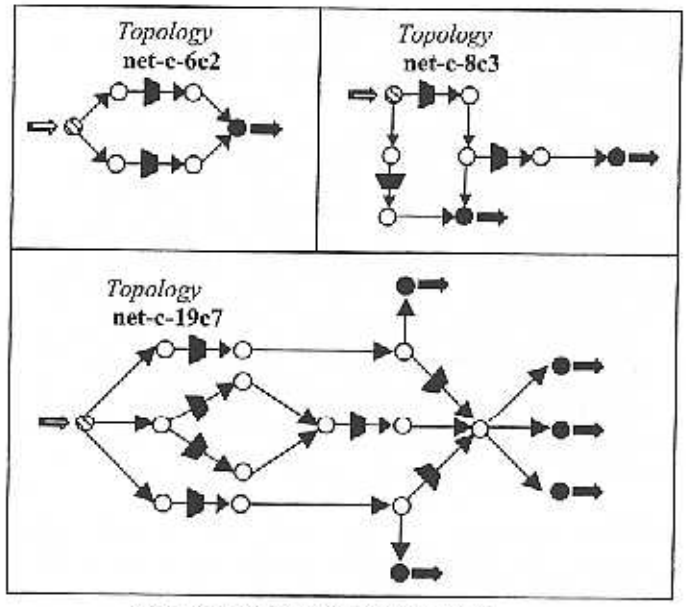

Figure 5. Examples of topologies type c used.

For example, in Fig. 5, a striped node (shown with an ingoing arrow next to it) represents a supply point, a black node (shown with an outgoing arrow next to it) represents a demand point, and a white node is a transshipment node. A single directed arc joining two nodes represents a pipeline, and a directed arc with a black trapezoid represents a compressor.

Our procedure was coded in C++, and run on a Sun Ultra 10 under Solaris 7. Computational evaluation was based for different mesh sizes (Δp=20, 5, 1). The computational results on gun barrel networks are shown in Table 1. The instances tested are shown in the first column; the following columns show the CPU time (sec) and objective function value for each of the mesh sizes testes. Similarly, computational results on tree and cyclic networks are shown in Tables 2 and 3, respectively.

Topology	Δp = 20 PSIG		Δp = 5 PSIG		Δp = 1 PSIG	
	CPU Time (sec)	Objective Value	CPU Time (sec)	Objective Value	CPU Time (sec)	Objective Value
net-a-5c1-C4	0.04	280,225.42	0.05	164,914.28	0.77	162,561.73
net-a-6c2-C1	0.07	2,305,101.07	0.17	1,925,665.87	6.48	1,802,576.73
net-a-6c2-C2	0.07	2,305,101.07	0.19	1,925,665.87	6.43	1,802,576.73
net-a-6c2-C3	0.04	1,065,709.48	0.29	399,126.31	15.03	343,249.56
net-a-6c2-C4	0.04	1,499,610.99	0.24	1,242,521.56	9.99	944,518.83
net-a-6c2-C5	0.05	3,650,144.05	0.51	3,259,430.35	21.77	3,044,941.55
net-a-6c2-C6	0.06	5,012,592.29	0.23	4,210,341.74	9.01	3,822,415.39
net-a-6c2-C7	0.03	1,269,300.15	0.16	694,097.81	3.39	558,114.25
net-a-6c2-C8	0.07	2,702,943.75	0.22	1,674,900.94	6.94	1,546,357.70
net-a-6c2-C9	0.01	5,045,312.43	0.07	4,066,445.48	1.77	4,012,593.02
net-a-8c3-C4	0.11	2,998,221.59	0.29	1,301,982.12	20.07	1,322,802.05
Topologies with special structures						
net-a-19c7-C4	0.07	4,652,595.76	0.91	3,340,553.26	39.31	3,046,053.04
net-a-12c1-C4	0.02	1,201,292.29	0.08	1,165,912.02	3.14	1,164,170.90
net-a-12c3-C4	0.05	4,085,543.34	0.49	3,334,569.62	24.31	3,213,108.46
net-a-18c1-C4	0.06	1,089,457.57	0.27	764,997.59	5.46	710,859.90
net-a-20c2-C4	0.06	3,018,255.30	0.76	2,261,313.45	40.64	2,099,562.15

Table 1. Computational results on gun-barrel networks.

Topology	Δp = 20 PSIG		Δp = 5 PSIG		Δp = 1 PSIG	
	CPU Time (sec)	Objective Value	CPU Time (sec)	Objective Value	CPU Time (sec)	Objective Value
net-b-10c3-C1	0.08	5,151,874.09	1.20	4,926,640.97	60.07	4,294,872.70
net-b-10c3-C2	0.10	5,151,874.09	1.13	4,926,640.97	57.12	4,294,872.70
net-b-10c3-C4	0.11	9,558,362.12	0.5	7,861,105.38	20.54	7,216,523.11
net-b-10c3-C5	0.08	6,323,834.73	1.00	5,651,798.53	42.38	5,191,111.05
net-b-11c1-C4	0.22	10,803,366.07	1.99	9,201,821.43	91.11	8,972,057.94
net-b-12c4-C2	0.04	3,132,246.81	0.37	2,917,123.65	2.43	2,854,285.65
net-b-12c4-C3	0.05	11,516,568.35	0.14	10,852,946.19	3.38	10,597,128.61
net-b-12c4-C4	0.05	2,984,542.04	0.20	2,684,886.41	9.77	2,258,057.33
net-b-15c6-C4	0.21	8,234,564.12	1.74	6,342,045.90	70.69	5,292,939.80
net-b-41c14-C1	0.30	28,952,769.30	2.94	24,198,619.10	161.3	22,223,410.34
net-b-41c14-C2	0.22	28,952,769.30	2.98	24,198,619.10	161.12	22,223,410.34
net-b-41c14-C3	0.20	37,244,343.96	2.10	34,386,871.29	114.71	32,475,771.32
net-b-41c14-C4	0.67	36,476,126.39	2.65	31,298,741.09	129.24	30,809,127.04

Table 2. Computational results on tree networks.

As we can see, the mesh size becomes an important factor in terms of accuracy of solution. We must point out that results for non-cyclic instances (Table 1 and 2) are global optimal, whereas results for the cyclic instances (Table 3) are "optimal" for the given flow. CPU times are very reasonable. For example, the test requiring more effort (Δp=1) never exceeded three minutes.

Topology	Δp = 20 PSIG		Δp = 5 PSIG		Δp = 1 PSIG	
	CPU Time (sec)	Objective Value	CPU Time (sec)	Objective Value	CPU Time (sec)	Objective Value
net-c-6c2-C1	0.03	2,279,711.77	0.16	1,884,060.48	3.24	1,852,252.53
net-c-6c2-C2	0.01	2,279,711.77	0.18	1,884,060.48	3.29	1,852,252.53
net-c-6c2-C3	0.02	1,200,668.26	0.20	1,041,265.76	4.07	1,020,842.29
net-c-6c2-C4	0.04	972,369.09	0.19	744,261.08	3.95	653,675.50
net-c-6c2-C5	0.02	2,999,798.61	0.19	2,603,372.66	3.36	2,716,611.07
net-c-6c2-C6	0.02	4,342,711.79	0.19	3,561,310.69	3.55	3,442,838.76
net-c-6c2-C7	0.001	1,082,391.42	0.15	850,674.93	2.95	788,218.02
net-c-6c2-C8	0.04	1,853,866.09	0.15	1,468,605.58	3.33	1,405,443.48
net-c-6c2-C9	0.03	1,499,524.36	0.17	1,294,608.69	2.95	1,243,179.03
net-c-10c3-C1	0.06	6,022,631.39	0.41	4,629,404.41	16.01	4,267,949.79
net-c-10c3-C2	0.04	6,022,631.39	0.38	4,629,404.41	15.32	4,267,949.79
net-c-10c3-C3	0.09	5,123,605.66	0.72	4,081,328.16	30.25	3,978,465.67
net-c-10c3-C4	0.05	6,076,716.15	0.31	4,652,223.35	7.31	4,360,146.30
net-c-10c3-C5	0.07	7,510,579.28	0.62	7,145,975.03	19.59	6,624,644.75
net-c-10c3-C6	0.11	11,908,871.1	0.61	10,996,571.93	15.51	10,766,001.9
net-c-10c3-C9	0.08	6,257,129.08	0.29	5,265,444.54	6.04	5,181,127.14
net-c-10c3-C9	0.12	4,223,456.43	0.48	3,315,770.23	11.17	3,201,774.65
net-c-15c6-C2	0.18	7,456,824.23	0.37	6,093,101.33	30.22	5,789,941.01
net-c-15c6-C4	0.18	3,143,098.07	0.45	2,612,943.12	18.43	2,154,914.02
net-c-18c7-C4	0.27	21,593,920.4	1.17	20,659,013.54	35.4	20,119,966.2

Table 3. Computational results on cyclic networks.

Finally, Table 4 shows the excellent behavior of the NDP algorithm against a GRG method [3] on cyclic networks. The instances tested are shown in the first column; immediately, the status (where, GS, LS, and IS mean global optimal, local optimal, and infeasible solution, respectively), CPU Time (sec) and the best objective value found are presented for each method in the analysis. First, the NDP was able to deliver solutions to all instances tested, whereas GRG failed for ten of these. NDP outperformed the GRG in terms of solution quality.

Topology	NDP Algorithm			GRG Method		
	Status	CPU Time (sec)	Objective Value	Status	CPU Time (sec)	Objective Value
net-c-6c2-C1	GS	3.24	1,852,252.53	LS	0.82	2,312,548.24
net-c-6c2-C2	GS	3.29	1,852,252.53	LS	0.82	2,312,548.24
net-c-6c2-C3	GS	4.07	1,020,842.29	LS	0.67	1,751,520.99
net-c-6c2-C4	GS	3.95	653,675.50	LS	1.01	1,393,061.12
net-c-6c2-C5	GS	3.36	2,716,611.07	LS	1.12	3,099,415.46
net-c-6c2-C6	GS	3.55	3,442,838.76	IS		
net-c-6n2-C7	GS	2.95	788,218.02	LS	0.57	988,098.79
net-c-6c2-C8	GS	3.33	1,405,443.48	LS	0.35	1,706,693.05
net-c-6c2-C9	GS	2.95	1,243,179.03	IS		
net-c-10c3-C1	GS	16.01	4,267,949.79	IS		
net-c-10c3-C2	GS	15.32	4,267,949.79	IS		
net-c-10n3-C3	GS	30.25	3,978,465.67	IS		
net-c-10c3-C4	GS	7.31	4,360,146.30	LS	1.03	5,610,632.12
net-c-10c3-C5	GS	19.59	6,624,644.75	IS		
net-c-10c3-C6	GS	15.51	10,766,001.9	IS		
net-c-10c3-C9	GS	6.04	5,181,127.14	IS		
net-c-10c3-C9	GS	10.72	3,281,774.65	IS		
net-c-15c6-C2	GS	30.22	5,789,941.01	LS	0.4	6,313,810.78
net-c-15c6-C4	GS	18.43	2,154,914.02	LS	0.18	3,566,353.60
net-c-19c7-C4	GS	35.4	20,119,966.2	IS		

Table 4. Behavior of the NDP algorithm against a GRG method.

6 Conclusions

Our computational results showed empirically how the problem structure can be efficiently exploited by taking advantage of a non-sequential dynamic programming technique. When using the finest discretization size, the computational effort never exceeded 3 minutes.

A central issue regarding the NDP algorithm applied to natural gas transmission network optimization is on how its performance, when compared with other methods on cyclic topologies, such as the GRG method, had more success on topologies that contains more compressor stations. Finally, the NDP Algorithm has been applied to several test instances representing dozens of different pipeline systems over a broad variety of flow conditions, with uniformly good results. So, this search algorithm not only found better solutions, but also reduced the resources (computational time) used by the computer. This represents a significant contribution, especially when dealing with cyclic structures where previous approaches had failed.

We must point out that results for non-cyclic instances are indeed global optimal, whereas results for the cyclic instances are "optimal" for the given flow. So, one current research trend is to develop a method to efficiently modify the flow values. The use of meta-heuristics such as GRASP or Tabu Search, whose internal mechanism for escaping local optima seems very attractive.

Acknowledgments: The research of the first author was supported by a fellowship for graduate studies from the Mexican National Council for Science and Technology (CONACYT). The research of the second author was supported by a research grant from CONACYT (grant J33187-A), and Universidad Autónoma de Nuevo León under its Scientific and Technological Research Support Program (PAICYT grants CA555-01 and CA763-02).

References:

[1] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, USA, 1957.

[2] R.G. Carter. Pipeline optimization: Dynamic programming after 30 years. In *Proceedings of the PSIG Meeting*, pages 1-19, Denver, USA, October 1998.

[3] H.J. Flores-Villarreal and R.Z. Ríos-Mercado. Computational experience with a GRG method for minimizing fuel consumption on cyclic natural gas networks. In N.E. Mastorakis, I.A. Stathopulos, C. Manikopoulos, G.E. Antoniou, V.M. Mladenov, and I.F. Gonos (editors), *Computational Methods in Circuits and Systems Applications*, pages 90-94. WSEAS Press, Athens, Greece, 2003.

[4] B.J. Gilmour, C.A. Luongo, and D.W. Schroeder. Optimization in natural gas transmission networks: A tool to improve operational efficiency. Technical report, Stoner Associates Inc., April 1989. Presented at the 3rd SIAM Conference on Optimization, 1989

[5] H.S. Lall and P.B. Percell. A dynamic programming based gas pipeline optimizer. In A. Bensoussan and J.L. Lions (editors), *Analysis and Optimization of Systems*, pages 123-132, Springer-Verlag, Berlin, Germany, 1990.

[6] R.Z. Ríos-Mercado. Natural gas pipeline optimization. In P.M. Pardalos and M.G.C. Resende (editors), *Handbook of Applied Optimization*, chapter 18.8.3, pages 813-825. Oxford University Press, New York, USA, 2002.

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

[8] Y. Villalobos-Morales, D. Cobos-Zaleta, H.J. Flores-Villarreal, C. Borraz-Sánchez, and R.Z. Ríos-Mercado. 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, Alabama, USA, January 2003.

[9] P.J. Wong and R.E. Larson. Optimization of natural gas pipeline systems via dynamic programming. *IEEE Transactions on Automatic Control*, AC-13(5):475-481, 1968.

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

[11] H.I. Zimmer. Calculating optimum pipeline operations. Technical Report, El Paso Natural Gas Company, 1975. Presented at the AGA Transmission Conference, 1975.