Proceedings of the 5th Annual World Conference
of the Society for Industrial and Systems Engineering,
San Francisco, CA, USA
October 13-14, 2016

Applying Exact Algorithms in a Problem of Manufacturing Network Flow

J. Meza-Calvillo[1], M. Cantú[2], R.J. Praga-Alejo[1,3], and D. González-González[1,3]

[1]Corporación Mexicana de Investigación en Materiales (COMIMSA), Calle Ciencia y Tecnología #790, Col. Saltillo 400, C.P. 25290, Saltillo, Coahuila, México. Phone: (+52) 01 844 411 32 00.

[2]Universidad Autónoma Agraria Antonio Narro, Calzada Antonio Narro #1923, Buenavista, CP. 25315, Saltillo, Coahuila, México. Phone: (+52) 01 844 411 02 09.

[3]Facultad de Sistemas, Universidad Autónoma de Coahuila, Ciudad Universitaria, Carretera a México Km. 13, Arteaga, Coahuila, México. Phone: (+52) 01 844 689 10 30.

Corresponding author's Email: julieta.meza@comimsa.com

Abstract: Nowadays networks have several applications in the industrial world; a very representative scenario is the manufacturing network. Delivering products on time and satisfying quality specifications is the main objective of a manufacture system within a supply chain, however ensuring the flow of raw materials for preventing the system to stop is another task it needs to assure. Sending specific goods through a system originates a scenario called network flow and the principle advantage of this flow is to facilitate the representation of different issues that occur within the manufacturing system. For these settings, exact algorithms are used to solve multiple problems that can be display; these algorithms find a set of potentials solutions, compare them and select the best one. This study focuses in 2 types of exact algorithms, the Dijkstra and Floyd-Warshall algorithms which comparisons are made, describing their advantages and disadvantages in a real-world-case.

Keywords: Exact Algorithms, Manufacturing Network Flow, Dijkstra, Floyd-Warshall.

1. Introduction

Networks are used to represent several kinds of scenarios, for example, in the electrical manner networks can model how many power they can send and how they distributed it in a balanced way. In the transportation subject it can be applied to rail roads, highways and airline services, to model the different pathways they can use to perform their schedule within a stablished time and cost. In the communication field, it has proved to be a very useful tool to communicate each other regardless the distance we face with minimum effort, just by picking the phone and dial the number. But the most complicated asset resides in the manufacturing sector, which provides all the commodities of food stock and consumer products that are used daily and additionally it is available 24 hours / 7 days of the week (Ahuja et al., 1993). For this matter, Fang and Qi (2003) proposed a specialized kind of network called manufacturing network flow, which represents better the dynamic of a manufacturing process, and how it can accomplish the complicated task of having all the products in time.

The manufacturing networks defer from the traditional distribution networks by some simple details. The most important is the kind of nodes it contains. These special nodes play an important part when we want to apply algorithms to an internal problem.

However, in manufacturing networks flow there are 2 general versions; the first one, refers to a distribution network which is defined by a scenario where there exist a supplier and wishes to send several amount of product into a certain network with several clients waiting for their shipment. The second one, refers to an assembly network, which is defined by an industrial process where raw materials enter the system and the result is a final product, whether this final product be cars, food, clothing, etc., (Fang & Qi, 2003).

Having this two general cases the focus of this paper will be in an assembly network. Assembly networks provides between 60 and 80% of the incomes of a country that specialized in this area, hence, many entities do not have the technology and the workforce to compete with first world countries like United States, Western Europe, Japan, etc., (Pardasani, 1991) (Sirkin et al., 2012). Having this detail into account, these entities somehow need to optimize their process and resources and by accomplishing this objective, the result would be a positive impact in their global economy (Samaranayake et al., 2011). Therefore, to develop strategies that can overall the performance of the network, exact algorithms can be of great help, they can provide an exact result from a specific type of problem (Skiena, 1998).

1.1 Problem Statement

In many scenarios in the industrial world, optimizing the resources is a key point to remain competitive in a globalized world. Many aspects within a company are tangible and it can be optimized, for example, the raw materials that the process will consume, how many machines will be used or how many workers will be in a shift, among other things. On the other hand, there also exist intangible aspects like time and economic resources.

Usually optimizing the time of a process can lead to greater developments within the system, however is quite a challenge regardless the size of the system. Nevertheless, accepting it may bring several benefits, a very visible profit can be reflected in the costs. Additionally, if it is decided to optimize the cost, might or might not optimize the time of process. For this matter it is needed to compare which of this cases will result better to implement.

So, the dilemma that a manufacturing industry has is, if the objective is to reduce the time of process, it is needed to analyzed whether the cost can be absorbed or not. If the cost is greater of what it was expected, then it would be wise to focus in reducing the time of process. To know which decision to make, it would be necessary a method that allows to know the cost-benefit of taking either of this decisions.

Having this point clear, it can change the scheduling of the process, for example, it could be cheaper if the company buy raw materials and process them to obtain a final product or perhaps would be prudent to buy sub products that conform the final product and only assemble them. On the other hand, having assemble all the raw materials would increase the time of process, however, having only assemble different sub products to achieve the establish goal may reduce the time of process but may increase the cost.

The principle objective is to find the shortest path inside the network that can provide an analysis that can help to make a decision regarding time or costs. Finding this path would contribute to have an idea how it could be optimized this resources. To obtain the shortest path in the network it would be applied exact algorithms.

Exact algorithms are useful when dealing with combinatorial optimization problems (Saurabh, 2007). These ensures that the problem will end with an exact answer, which has the advantage of being applied in theory and practical scenarios (Gerhard, 2002). So, for this matter, it is proposed to implement 2 exact algorithms: Dijkstra and Floyd-Warshall. These two provide the answer of a shortest path within any network, with a very slight difference in their development.

2. Methodology

2.1 Shortest Path Problem

The shortest path has a wide variety of applications and it appears implicitly in many scenarios. Some of this applications may result in finding a path between two specified locations (nodes) that may contain a minimum length if there is a necessity to know distances, time is also important, may be the top priority to know which path to take that makes the least time to a specified destiny or if the objective is to know the costs, it can also provide which path will be cheaper to travel. All these scenarios bring the following programming formulation (Ahuja et al., 1993):

$$Minimize \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1}$$

$$subjet\ \ to:$$

$$\sum_{\{j:(i,j)\in A\}} x_{ij} \; - \sum_{\{j:(j,i)\in A\}} x_{ji} = \begin{cases} n-1 & for \;\; i = s \\ -1 & for \;\; all \;\; i \in N - \{s\} \\ 0 & otherwise \end{cases} \tag{2}$$

$$x_{ij} \geq 0 \qquad \forall\, (i,j) \in A \tag{3}$$

Where equation (1) is the objective function to minimize the cost of all the flow in an arc (i,j). Constraint (2) exemplified the shortest path to choose within the network, finally constraint (3) represents that all the flow cannot be negative.

2.2 Methods

To find an optimal path, either the longest, shortest, cheapest or expensive one, is quite a complicated task (Fan et al., 2005). The shortest path arises in several networks problems and shares the characteristic of being a sub problem in bigger problems of network flow (Xu et al., 2007). This problem is the simplest and the basic idea lies in finding a way, the shortest way, between two points (Ahuja et al., 1995). In this case it will find the shortest path between the source node (where raw materials input the system) and the sink node (where the final product will be). In case it is decided to use sub products the node source and node sink will change. For this task it will be employed the Dijkstra and Floyd-Warshall algorithms.

2.2.1 Dijkstra Algorithm

The Dijkstra algorithm is a classic in network flow problems and a basic algorithm (Misra, 2001). This algorithm was first implemented by Edsger Dijkstra in 1959, the finality was to find the shortest path from a source node (denoted with the letter s) to a sink node (denoted with the letter t). The procedure of solution is the following:

In a directed network $G = (N, A)$ where N is a set of nodes and A a set of arcs, it is obtained the adjacency node to node matrix where it will be stock the distances, costs or times that exist between each node, this matrix will be called Omega (Ω). To begin the algorithm is necessary to take the first row of Ω, this row will represent the weight vector (D_w). Then it would be necessary to select a pivot element which is the lowest weight in D. Once the pivot element (D_p) is selected, it compares with each weight in D_w with the following formula: $D_p < D_w + \Omega\,(i, j)$. After the comparison is finished, the result would be a second vector which is a set of weights of the remaining nodes. To choose the next node, the algorithm repeats the previous steps and so on, until it reaches the sink node. At the end, once it achieves the sink node the algorithm provides the total length, time or cost that the shortest path has (Torrubia & Terrazas, 2012).

2.2.2 Floyd-Warshall Algorithm

The Floyd-Warshall algorithm is a generalization of Dijkstra's algorithm and is said to be more flexible than the previous one, for that reason it also said to be a great example of dynamic programming. This algorithm was published in 1959 by Bernard Roy, later on, in 1962 several researchers publish similar algorithms. The principle idea of this algorithm is to find the shortest path within a network having any node to be the node source and sink. The procedure of solution is the following:

In a directed network $G = (N, A)$ where N is a set of nodes and A a set of arcs, it is obtained 2 matrixes: the first one is a matrix of distance (D_o), which is an adjacency node to node matrix, and the second is a sequence matrix (S_o). The number of iterations that this algorithm will perform corresponds with the number of nodes that the network has, for this purpose, each iteration is represented with the letter k. As it starts with iteration 1 ($k = 1$), the algorithm selects the first row and column of each matrix and applies the formula $d_{ij} > d_{ik} + d_{kj}$ (where i is consider to be a row and j a column) to the rest of the other positions in the matrixes. When all the positions are calculated, the algorithm checks if the conditions are met; if this is true the positions will change their value, the S_1 matrix will take the number of the iterations in which the algorithm is and D_1 matrix will adopt the smallest digit in the inequalities. If the conditions are not met, the positions will hold the original digits of the matrixes D_o and S_o. Once finished, the algorithm provides complete matrixes D_f and S_f, where D_f contains the shortest distances between every node in the network and S_f has the paths that it could be transit to get to every node in the system (Sen et al., 2013).

3. Application

An application of these two exact algorithms is performed, having an assembly network and the following data presented in Table 1 concerning the same system:

Table 1. Time and Costs of an Assembly Network with Raw Materials

Arcs	Time *(seconds)*	Costs *(cents)*
(1,2)	22.77	15.72
(1,3)	9.24	45.89
(1,4)	10.80	20.40
(1,5)	46.45	13.65
(2,6)	47.06	16.05
(2,7)	30.31	32.11
(2,9)	7.63	25.82
(3,6)	15.33	20.47
(3,8)	20.53	41.56
(4,6)	41.13	30.75
(4,8)	5.67	29.19
(4,9)	6.89	45.36
(5,7)	12.43	17.58
(5,8)	33.56	38.40
(6,10)	37.19	38.32
(6,11)	33.50	38.16
(7,10)	29.06	21.74
(7,11)	24.84	29.98
(8,10)	18.03	8.34
(9,10)	37.76	7.37
(10,11)	91.27	97.65

The Dijkstra and Floyd-Warshall algorithms were applied taking into consideration that the system will work with raw materials, both algorithms obtain the same results in time and cost. This comparison is not a problem, both algorithms have found the shortest path in the network. Results are shown in Table 2. In the first run of the experiment, it obtained that the shortest path giving priority to the aspect of *Time* can be very expensive. On the other hand, when the priority are the *Costs*, the time is not that high. For this reason, the results are 2 paths that the production plan can consider to achieve a better use of resources.

Table 2. Results obtained with Raw Materials

Method	Giving Priority to Time			Giving Priority to Costs		
	Time *(seconds)*	Costs *(cents)*	Route Time	Costs *(cents)*	Time *(seconds)*	Route Costs
Dijkstra Algorithm	58.07	104.52	$1 \rightarrow 3 \rightarrow 6 \rightarrow 11$	61.21	83.72	$1 \rightarrow 5 \rightarrow 7 \rightarrow 11$
Floyd-Warshall Algorithm	58.07	104.52		61.21	83.72	

However, to carry out a comparison between using raw materials or sub products, it would be necessary to perform a second run of the experiment, where the network will not have a specified node source. Obtaining this data, it can be compared how much time and money it consumes the shortest path of the modified system. The modified data will be presented in Table 3 and their results in Table 4.

Table 3. Time and Costs of an Assembly Network with Subproducts

Arcs	Time *(seconds)*	Costs *(cents)*
(1,5)	47.06	16.05
(1,6)	30.31	32.11
(1,8)	7.63	25.82
(2,5)	15.33	20.47
(2,7)	20.53	41.56
(3,5)	41.13	30.75
(3,7)	5.67	29.19
(3,8)	6.89	45.36
(4,6)	12.43	17.58
(4,7)	33.56	38.40
(5,9)	37.19	38.32
(5,10)	33.50	38.16
(6,9)	29.06	21.74
(6,10)	24.84	29.98
(7,9)	18.03	8.34
(8,9)	37.76	7.37
(9,10)	91.27	97.65

Table 4. Results obtained with Sub products

Method	Giving Priority to Time			Giving Priority to Costs		
	Time *(seconds)*	Costs *(cents)*	Route Time	Costs *(cents)*	Time *(seconds)*	Route Costs
Dijkstra Algorithm	37.27	47.56	$4 \rightarrow 6 \rightarrow 10$	47.56	37.27	$4 \rightarrow 6 \rightarrow 10$
Floyd-Warshall Algorithm	37.27	47.56		47.56	37.27	

Since the modifications had been made, it could be seen that both algorithms obtain the same results, although this time the difference is that the optimal route do not prioritize *Time* or *Costs*. If the decision is made to prioritize *Time* would be same if the choice could have been to prioritize *Costs*. Also, this provides a unique path that can achieve either of this goals.

4. Conclusions

It is shown the comparison of two exact algorithms which provides an analysis for taking decisions of working with raw materials or sub products. In this particular case it has been shown 2 important aspects.

The first one, implementing exact algorithms can be useful when the scenario requires an exact answer. In this case, it offers promising results, which provides data that can be useful for taking a decision in the assemble program. Working with sub products may be a better option, taking into account that the path provided by the two algorithms is the same, either prioritizing time or costs. Unlike working with raw materials would require to make a choice between this two options.

The second aspect focus in the dynamic of both algorithms. Not only exist a requirement to optimize the resource that a company has, but also a great contribution may lie optimizing the methods that will be used to solve the problem. The problem described earlier have two different methods to choose upon, Dijkstra and Floyd-Warshall; the first one is a classic algorithm of network flow problems, but has the limitation that it only shows the shortest path between 2 previously defined nodes, which means that it has to perform several iterations to discover the shortest path between all the combinations of nodes.

On the other hand, Floyd-Warshall algorithm since the first iteration provides two final matrixes, one of distances (D_f) and one of sequences (S_f), together they provide the shortest path between every single node within the system. This option gives an important advantage versus the Dijkstra algorithm, that is why, Floyd-Warshall is better in this type of scenarios.

For future work, it would be to employ different exact algorithms with another assembly network contemplating diverse aspects excluding time and costs. Another point to consider would be applying metaheuristics methods in this scenarios to observed the performance of such methods and to compare the obtain results between them and with the exact methods.

5. References

Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. United States: Prentice Hall Inc.

Ahuja, R. K., Magnanti, T. L., Orlin, J. B. & Reddy, M. R. (1995). Applications of Network Optimization. *Handbooks in Operations Research and Management Science*, 7 (1), 83.

Fan, Y. Y., Kalaba, R. E., & Moore, J. E. (2005). Shortest Paths in Stochastic Networks with Correlated Link Costs. *Computers & Mathematics with Applications*, 49, 1549-1564.

Fang, S. C. & Qi, L. (2003). Manufacturing Network Flows: A Generalized Network Flow Model for Manufacturing Process Modelling. *Optimization Methods and Software*. 18, 143-165.

Gerhard, J. (2002). Exact Algorithms for NP-hard Problems. *OPTIMA Mathematical Programming Society Newsletter N*, 68.

Huang, K. (2011). *Maximum Flow Problem in Assembly Manufacturing Networks*. North Carolina: North Carolina State University.

Matuschke, J. (2014). *Network Flows and Network Design in Theory and Practice*. Berlin: Mathematik und Naturwissenschaften der Technischen Universität Berlin

Misra, J. (2001). A Walk Over the Shortest Path: Dijkstra's Algorithm Viewed as Fixed-Point Computation. *Information Processing Letters*, 77, 197-200.

Pardasani, A. (1991). *Network Flow Optimization Models for Integrated Flexible Manufacturing Systems*. Ottawa, Ontario: Carleton University.

Samaranayake, P., Laosirihongthong, T., & Chan, F. T. (2011). Integration of Manufacturing and Distribution Networks in a Global Car Company–Network Models and Numerical Simulation. *International Journal of Production Research*, 49, 3127-3149.

Saurabh, S. (2007). *Exact Algorithms for Optimization and Parameterized Versions of some Graph Theoretic Problems*. Mumbai, India: Homi Bhaba National Institute

Sen, S. K., Gupta, S., & Mukhopadhyay, I. (2013). Economic Viability of an Alternative Internal Road Network in Tripura: An Application of Shortest Path Algorithm. *Hill Geographer*, 29, 25-39.

Sirkin, H. L., Rose, J., & Zinser, M. (2012). *The US Manufacturing Renaissance: How Shifting Global Economics Are Creating an American Comeback*. United States.

Skiena, S. S. (1998). *The Algorithm Design Manual.* United States, New York: Springer Science & Business Media.

Torrubia, G. S., & Terrazas, V. L. (2012). *Algoritmo de Dijkstra. Un tutorial interactivo*. Valencia, España: VII Jornadas de Enseñanza Universitaria de la Informática

Xu, M. H., Liu, Y. Q., Huang, Q. L., Zhang, Y. X., & Luan, G. F. (2007). An Improved Dijkstra's Shortest Path Algorithm for Sparse Network. *Applied Mathematics and Computation*, 185, 247-254.