ALLERTON CONFERENCE ON COMMUNICATIODN, CONTROL AND COMPUTRS, SEPTEMBER 1987

AN EFFICIENT ASYNCHRONOUS DISTRIBUTED PROTOCOL TO TEST FEASIBILITY OF THE DUAL TRANSSHIPMENT PROBLEM

M. A. COMEAU
Centre de recherche informatique de Montréal
Montréal (Canada)

K. THULASIRAMAN
Dept. of Electrical and Computer Engineering
Concordia University, Montréal (Canada)

K. B. LAKSHMANAN
Dept. of Computer Science
Concordia University, Montréal (Canada)

ABSTRACT

An asynchronous distributed protocol to test feasibility of the dual transshipment problem is presented. The message and time complexities of this protocol are $O(mn)$ and $O(n)$, respectively, where m and n are the number of edges and the number of nodes in the graph underlying the problem. This protocol can be easily modified to construct a starting solution required to initialize the primal-dual method. A mechanism is incorporated in these protocols to detect infeasibility, whenever it occurs.

I. INTRODUCTION

Network optimization refers to the class of optimization problems defined on graphs and networks. In recent years, efficient distributed protocols or algorithms have been obtained for several network optimization problems such as the single-source shortest-path problem, the maximum-flow problem, the problem of constructing a maximum matching, etc. In this paper, we are concerned with certain issues in the design of distributed protocols for the transshipment problem [1] which generalizes several of the network optimization problems such as the ones stated above.

The transshipment problem can be formulated as a linear program and therefore can be solved by the simplex method using any one of the three approaches, namely, the primal, the dual or the primal-dual methods. The network simplex algorithm is an efficient implementation of the primal method for solving the transshipment problem and has been extensively studied in the literature. However, the dual method has received very little attention, though one can construct an optimum solution of the problem from an optimum solution of the dual. Our recent investigations [2] indicate that the network simplex algorithm is essentially sequential in nature and does not offer much scope for an efficient distributed implementation. This has motivated us to select the dual and the primal-dual methods as candidates in our study of issues in distributed protocol design for network optimization problems.

The simplex method involves two phases. In the first phase, feasibility of the given problem is tested and a basic feasible solution is constructed, if such a solution exists. The second phase starts with a basic feasible solution and through a sequence of pivot operations constructs an optimum solution, if available.

In this paper, we will be primarily concerned with the first phase of the dual transshipment problem. Thus, we are interested in designing an efficient distributed protocol to test feasibility of this problem. In Section II, we present the dual transshipment problem and present the important features of a sequential algorithm to test feasibility. This algorithm is of time complexity $O(mn)$ where m and n are, respectively, the number of edges and the number of nodes of the graph underlying the dual transshipment problem. In Section III, we show that the algorithm of Section II can be suitably modified to construct a starting solution required to initialize the primal-dual method. In Section IV, we present the model we use for distributed computation and then present, in Section V, the essential features of a synchronous distributed protocol for the feasibility testing problem. This protocol is of message complexity $O(mn)$ and time complexity $O(n)$. Combining this synchronous protocol with the α-synchronizer of Awerbuch [3],[4] or with the simpler synchronizer presented in [10] leads to an asynchronous protocol whose message and time complexities are the same as those of the synchronous protocol.

This research was supported in part by Concordia University under grant CASA-N67 and by FCAR Québec under grant 87AS2407 of the "Actions Spontanées" program.

II. ALGORITHM FEASIBLE: TESTING FEASIBILITY OF THE DUAL TRANSSHIPMENT PROBLEM

Given a weighted graph $G = (V,E)$ on n nodes and m edges, with the weight vector M_0 associated with the edge set E, let A denote the incidence matrix of G. Then the ***dual transshipment*** problem is a linear program defined as follows:

$$\textit{minimize } \Omega\Sigma$$

$$\textit{subject to } A^t\Sigma \geq -M_0 \tag{1}$$

$$\Sigma \geq 0 \tag{2}$$

where Σ is a column vector of dimension n and Ω is a row vector also of dimension n. Vectors Ω and M_0 are specified.

A vector $\Sigma \geq 0$ which satisfies (1) is called a ***feasible*** solution of the dual transshipment problem. If such a vector exists, then the given dual transshipment problem is feasible.

Theorem 1:

The dual transshipment problem is feasible if and only if the weighted graph G has no directed circuit of negative weight. //

Let d_{ij} denote the weight of a minimum-weight directed path from node i to node j. Let

$$\gamma_i = \max\{0, -\min_j\{d_{ij}\}\}, \; i = 1,2,\cdots,n. \tag{3}$$

Theorem 2:

The vector $\Sigma = (\gamma_1,\gamma_2,\cdots,\gamma_n)^t$ is a feasible solution of the dual transshipment problem if the weighted graph G has no directed circuit of negative weight. //

Proof of the above theorems may be found in [11].

To present an algorithm which obtains the feasible solution given in the above theorem, let us define the ***node firing*** operation as follows. Firing x times a node v refers to the operation of adding x to the weight of every outgoing edge at v and subtracting x from the weight of every incoming edge at v. In the following, $M(e)$ denotes the weight of edge e.

Algorithm FEASIBLE:

1. Let $M = M_0$.
2. While there exists an edge $e = (i,j)$ such that $M(e) < 0$, fire node i, $-M(e)$ times, updating M. //

Theorem 3:

If the weighted graph has no negative-weight directed circuits under the weight vector M_0, then Algorithm FEASIBLE terminates in a finite number of steps after firing each node i exactly γ_i times. //

Proof of correctness and termination of this algorithm may be found in [11].

We next present an implementation of the above algorithm which achieves a time complexity of $O(mn)$. The main steps in this implementation are:

Step 1: For each i, set $\gamma_i = 0$. Let M be the initial weight vector associated with the edge set.

Step 2: For each i, compute

$$\sigma_i = \max\{0, -\min_j\{M(i,j)\}\}$$

Step 3: Fire each node i, σ_i times and update γ_i by adding σ_i to its current value (Note that firing results in updating M also).

Step 4: If $\sigma_i = 0$ for all i, STOP. ELSE return to Step 2.

We now proceed to prove that the above implementation of Algorithm **FEASIBLE** achieves a complexity of $O(mn)$. Our proof also shows that this implementation computes γ_i's correctly whenever there is no circuit of negative weight in G.

In the following, we shall refer to one execution of Step 2 and Step 3 as a ***sweep***. Also, whenever there is no negative-weight directed circuits, it can be proved that the graph has at least one node i with $\gamma_i = 0$. Such a node will be referred to as a ***datum*** node. Note that in a graph, there may be more than one datum node. It can be proved that Algorithm **FEASIBLE** will never fire a datum node. A directed path of weight $-\gamma_i$ will be denoted by $p_{i,i'}$ where i' is clearly a datum node. We refer to the number of edges in a path as the ***length*** of the path.

Theorem 4:

Assume that G has no negative-weight directed circuits under the initial weight vector M_0. Consider any node i for which $\gamma_i > 0$. If the length of the path $p_{ii'}$ is k, then the node i will have been fired γ_i times at the end of the k^{th} sweep in the above implementation of **FEASIBLE**.

Proof:

Let $p_{ii'} = i, i_1, i_2, \cdots, i_{k-1}, i'$. At the beginning of the first sweep, the weight on the edge (i_{k-1}, i') is $-\gamma_{ik-1}$. Also, i' is never fired. So at the end of the first sweep, node i_{k-1} will have been fired γ_{ik-1} times. At the beginning of the second sweep, i_{k-1} is a datum in the new weight pattern and the weight on the edge (i_{k-2}, i_{k-1}) will be $-\gamma_{ik-2}$. So, during this sweep, node i_{k-2} will be fired γ_{ik-2} times. Repeating these arguments, we can see that at the end of the k^{th} sweep, node i will have been fired γ_i times. //

Theorem 5:

If G has no negative-weight directed circuits under M_0, then Algorithm **FEASIBLE** will terminate in no more than n sweeps, where n is the number of nodes in G.

Proof:

Each directed path $p_{ii'}$ in G is of length $\leq n-1$. So, by Theorem 4, each node i will be fired a total of γ_i times in no more than $n-1$ sweeps and the theorem follows. //

During each sweep, $O(m)$ edges are examined. Thus, each sweep takes $O(m)$ time and, hence, we have the following theorem.

Theorem 6:

The complexity of Algorithm **FEASIBLE** is $O(mn)$ if G has no negative-weight directed circuits under the initial weight pattern M_0. //

Any $\sigma_i > 0$ after the n^{th} sweep of Algorithm **FEASIBLE** indicates the presence of a negative-weight directed circuit. We can also incorporate in this algorithm a mechanism to detect the edges in such a circuit. Algorithm **FEASIBLE** can be suitably modified to handle the general case where there are upper as well as lowerbounds specified on the firing numbers.

III. INITIALIZATION OF THE PRIMAL-DUAL METHOD

The ***upperbounded*** transshipment problem is as follows:

$$\begin{aligned} &\textit{minimize } cx \\ &\textit{subject to } Ax = b,\ 0 \leq x \leq u. \end{aligned} \tag{4}$$

To initialize the primal-dual method, one is required to determine a set of x_{ij}'s and γ_{ij}'s such that

1. $0 \le x_{ij} \le u_{ij}$ (5)

2. $x_{ij} = 0$ whenever $\gamma_i + c_{ij} \ge \gamma_j$ (6)

3. $x_{ij} = u_{ij}$ whenever $\gamma_i + c_{ij} < \gamma_j$ (7)

It can be seen [2] that the initialization problem reduces to finding γ_i's such that

$$\gamma_i - \gamma_j + c_{ij} \ge 0 \tag{8}$$

whenever $u_{ij} = \infty$. If such γ_i's can be found then we can set

$$x_{ij} = \begin{cases} 0 & \text{if } \gamma_i + c_{ij} \ge \gamma_j \\ u_{ij} & \text{if } \gamma_i + c_{ij} < \gamma_j \end{cases} \tag{9}$$

Suppose we modify Algorithm **FEASIBLE** so that edges for which $u_{ij} < \infty$ are ignored while computing σ_i. Then we can see that such a modified algorithm will compute correctly a set of γ_i's (whenever such a set exists) satisfying (9).

IV. MODEL OF DISTRIBUTED COMPUTATION

For synchronous computations, we follow the model used in [3] - [5] and for asynchronous computations, we follow the model used in [3], [4], [6] - [9]. These models are by far the most commonly used ones. It might be pointed out that in these models it is necessary that all the messages received at a processor are transferred to a single common queue before being processed one by one. It is also assumed that the actions necessary for processing a message can all be performed in negligible computation time, without wait once started, and also uninterrupted by the arrival of other messages. Note also that in the asynchronous model, the communication subsystem is assumed to deliver a message to its destination after *a finite but, undetermined time lapse.* On the other hand, in a synchronous network model, we assume the existence of a global clock, so that all messages are sent only when a clock pulse is generated. Moreover, a message sent by any processor to its neighbor arrives at its destination before the next pulse is generated.

It must be emphasized that in all our protocols, no processor is assumed to be aware of the number of processors in the system.

As regards complexities, the message complexity of a distributed protocol is the total number of messages transmitted during the execution of the algorithm. The time complexity (both in the synchronous and the asynchronous cases) is the time that elapses from the beginning to the termination of the algorithm, assuming that the delay in any link is exactly one unit of time. Note that this assumption of unit delay in links, even for the asynchronous case, is made only for the purpose of timing analysis.

V. A SYNCHRONOUS DISTRIBUTED PROTOCOL

We now present the essential features of a synchronous distributed protocol for testing feasibility of the dual transshipment problem. This protocol is given in two phases. In the following, we shall assume that each node processor v is aware of the weights of all the outgoing edges at v.

After the usual **WAKE-UP** protocol, a leader s is first elected and a tree T rooted at s is constructed. The root s then informs all the nodes to start Phase 1. (Note that leader election requires $O(m + n \log n)$ messages and $O(n)$ time [12]).

Phase 1:

To start with, the firing number γ_w of each node w is set to zero.

Two types of messages -WAVE and ACK- are used. During the first pulse of activity each node processor w sets itself to the **START** state, sets PRED(w) = w and computes the minimum of the weights of all the outgoing edges incident at$w$. Let this minimum be ∂_w. If $\partial_w > 0$ then node w changes its state to

SUCCESS. Otherwise, it updates the weight of each outgoing edge by adding $|\partial_w|$ to its current weight and updates its firing number by adding $|\partial_w|$ to the current value of γ_w. Also, a WAVE message carrying the value $|\partial_w|$ is broadcast along each incoming edge at w.

During each subsequent pulse the node w processes the messages received, one at a time. Consider the WAVE message received along the i^{th} outgoing edge. Let FIRE(i) denote the value carried by the message. Now node w updates the weight of the i^{th} outgoing edge by adding -FIRE(i) to its current weight. Let MIN(w) denote the minimum of the current weights of all the outgoing edges at w after all the WAVE messages have been processed.

If MIN(w) ≥ 0, then node w sends an ACK message to every neighbor from which a WAVE message has been received. If MIN(w) < 0, then node w updates its firing number by adding |MIN(w)| to its current value. The weight of every outgoing edge is updated by adding |MIN(w)| to the current weight and then an ACK message is sent to PRED(w) if PRED(w) $\neq w$. Node w then arbitrarily selects an edge, say the edge (w,k) whose current weight is zero, sets PRED(w) = k, and sends ACK messages for all the WAVE messages processed, except for the one which was received along the edge (w,k). Finally, a WAVE message which carries the value of |MIN(w)| is sent along each incoming edge. If no incoming edge is present at w, an ACK message will be sent to PRED(w).

Note that at any time an ACK message will be pending on no more than one outgoing edge incident at any node. When node w has received ACK messages for all the messages it has sent, it transmits an ACK message to PRED(w).

Node w sets its state to **SUCCESS** when it has received ACK messages for all the messages it has sent during its first pulse.

When a node w and all its descendants in the tree T have reached the SUCCESS state, node w informs its father in T accordingly. When the leader s recognizes that all the nodes including itself are in the SUCCESS state, it initiates Phase 2.

Phase 2:

The objective of Phase 2 is to test for the presence of a directed circuit of negative weight. This is done by the leader transmitting messages along T and checking if at any node an ACK message is pending on an outgoing edge. If so, it indicates the presence of a negative-weight directed circuit, and all the nodes are then switched to the **INFEASIBLE** state (indicating infeasibility of the problem.) Otherwise, all the nodes are switched to the **FINISHED** state; in this case the current firing numbers of the nodes give a feasible solution for the given dual transshipment problem.

VI. CORRECTNESS AND COMPLEXITY OF THE SYNCHRONOUS PROTOCOL

To prove the correctness and establish the complexity of the synchronous protocol, first note that the actions taken by the nodes during any pulse of the first phase are essentially the same as those during the corresponding sweep of the sequential Algorithm **FEASIBLE**. Thus, if we assume that there are no negative-weight directed circuits, then the firing numbers of all the nodes would reach the values as given by (3) in no more than n pulses and hence, the weights of all edges would attain nonnegative values within this period. No WAVE messages will flow in the graph after the n^{th} pulse. The crucial problem then is to prove the correctness of the mechanism we have used to let every node know that all the edges have attained nonnegative weights and that the objective of the protocol has been achieved.

To do so, let us first assume that there are no negative weight directed circuits and that all the edges with negative weights are incident out of a single node w. Let the edge having the most negative weight be (w,i). Thus, during the first pulse of the first phase, node w is the only one which initiates a WAVE message. Any node x initiating a WAVE message during the first pulse will be considered as initiating the x-wave. The purpose of the w-wave is to fire node w so that the weight of the edge (w,i) becomes zero. However, this firing may cause weights of some of the edges incident into node w to become negative and thus will, in turn, trigger new WAVE messages (in fact w-wave messages). Let G_w denote the set of all edges traversed by the w-wave messages.

Note that a WAVE message received at, say, node j, along edge (j,k) is acknowledged either i) when it triggers no new WAVE messages or ii) when all the WAVE messages it triggered have been acknowledged or iii) when a later WAVE message arriving at j causes further firing of node j. In all these cases, all the edges traversed by the w-wave message received at node j along edge (j,k) would have attained nonnegative values. This means that when node w has received ACK messages for all the messages it sent during the first pulse, all edges in G_w would have attained nonnegative values. Therefore, at this point, node w detects completion of the mission of the w-wave message and sets itself to the SUCCESS state.

In the general case, more than one node may initiate a wave during the first pulse. The subgraphs traversed by the waves may not be distinct. When all the edges in any one of these subgraphs have attained nonnegative weights, the node initiating the corresponding wave would set itself to the SUCCESS state. When all the nodes reach the SUCCESS state, the root node can detect this and in the second phase it will inform all the nodes about the successful termination of the protocol.

Consider next, the case when there are some negative directed circuits. Suppose a node, say j, in such a circuit sends a WAVE message along edge (k,j) during the first pulse. Since j is in a negative-weight directed circuit and the length of a circuit is at most n, this node will be fired again within the first n pulses, resulting in a WAVE message along (k,j). This, in turn, sends an ACK message for the message sent along (k,j) during the first pulse and a new ACK message will be pending on (k,j). Thus node j will be able to set itself to the SUCCESS state within the first n pulses. The new pending ACK message on (k,j) indicates the presence of a negative-weight directed circuit and will be detected during the second phase. It can be shown that an ACK message will be pending at at least one node in a negative-weight directed circuit.

It is clear from the above discussion that all nodes will reach the SUCCESS state within the first n pulses. The root will detect this in $O(n)$ pulses. The second phase will terminate in $O(n)$ pulses. Furthermore, the number of messages during each pulse is $O(m)$. As pointed out above, leader election can be accomplished using $O(m + n \log n)$ messages and $O(n)$ time. Thus, we have the following theorem.

Theorem 7:
A feasible solution, if it exists, to a dual transshipment problem defined on a network of m edges and n nodes can be computed using a synchronous distributed protocol which requires $O(mn)$ messages and $O(n)$ time. //

VI. AN ASYNCHRONOUS DISTRIBUTED PROTOCOL

Combining the synchronous protocol of the previous section with the α-synchronizer of Awerbuch [3], [4] or the simpler synchronizer presented in [10], an asynchronous distributed protocol requiring $O(mn)$ messages and $O(n)$ time can be constructed for testing feasibility of the dual transshipment problem. Note that the synchronizer is required only for Phase 1 of this protocol.

VII. CONCLUSIONS

We have shown that an asynchronous distributed protocol of message complexity $O(mn)$ and time complexity $O(n)$ can be constructed to test feasibility of a dual transshipment problem defined on a network of m edges and n nodes.

VII. REFERENCES

[1] V. Chvátal, "Linear Programming", Freeman Company, San Francisco, 1983.

[2] Marc Comeau, "Reachability and Sequencing in Marked Graphs and State Graphs: Algorithms Based on Network Programming", Ph.D. thesis, Dept. of Electrical Engineering, Concordia University, June 1986.

[3] B. Awerbuch, "Complexity of Network Synchronization", J. Assoc. Comput. Mach., Vol. 32, No. 4, Oct. 1985, pp. 804-823.

[4] B. Awerbuch, "Reducing Complexities of the Distributed Max-Flow and Breadth-First-Search Algorithms by means of Network Synchronization", Networks, Vol. 15, 1985, pp. 425-437.

[5] E. Korach, D. Rotem and N. Santoro, "Distributed Algorithms for finding centers and medians in Networks", ACM Trans. Prog. Lang. Systems, Vol. 6, No. 3, July 1984, pp. 380-401.

[6] K. M. Chandy and J. Misra, "Distributed Computation on Graphs: Shortest-path Algorithms", Comm. Assoc. Comput. Mach. Vol. 25, No. 11, Nov. 1982, pp. 425-437.

[7] R. G. Gallager, "Distributed Minimum Hop Protocols", Technical Report LIDS-P-1175, Massachusetts Institute of Technology, U.S.A., Jan 1982.

[8] K. B. Lakshmanan, N. Meenakshi and K. Thulasiraman, "A Time-Optimal Message Efficient Distributed Algorithm for Depth-First-Search", Info. proc. Letters, Vol. 25, No. 2, May 1987.

[9] A. Segall, "Distributed Network Protocols", IEEE Trans. Info. Theory, Vol. IT-29, No. 1, Jan 1983, pp. 23-35.

[10] K. B. Lakshmanan and K. Thulasiraman, "On the Use of Synchronizers for Asynchronous Communication Networks", Proceedings of the 2nd International Workshop on Distributed Algorithms, Amsterdam, July 1987.

[11] M. A. Comeau and K. Thulasiraman, "Structure of the Submarking Reachability Problem and Network Programming", To appear in the IEEE Transactions on Circuits and Systems.

[12] B. Awerbuch, "Optimal Distributed Algorithms for Minimum Weight Spanning Tree, Counting, Leader Election and Related Problems", Proc. 19th Annual ACM Symp. on Theory of Computing, New York City, May 1987.