A PARALLEL OPTIMAL ROUTING ALGORITHM †

Cassilda Ribeiro †† and Didier El Baz
LAAS du CNRS
7, avenue du Colonel Roche, 31077 Toulouse Cedex France

Abstract

In this paper we propose a new approach for solving optimal routing problems in packet-switched networks, a particular class of multicommodity convex network flow problems. The method developed here is designed to obtain good rate of convergence while maintaining algorithmic simplicity and making effective use of parallel computing facilities.

1. Introduction

In a packet-switched communication network, messages sent by computers are segmented into short bit strings called packets. Packets are transmitted through the network as individual entities. When packets arrive at a node, they wait in queues for their turn to be transmitted on the next link in their path. Packets are reassembled into messages at the destination. In general a number of different paths are available for the flow of packets. Thus an important problem in packet-switched communication networks is the routing problem. This problem consists of obtaining an assignment of routes to the packets which is optimal according to some cost criterion. Minimum average message delay is the most frequently used criterion in the literature.

† Part of this work has been supported by Stimulation Action Contract SCI. 0047.C(H).
†† This author is on leave from Instituto de Ciencias Matematicas de Sao Carlos, USP, Brasil.

The formulation of a mathematical model for the routing problem was given by Kleinrock ([12], see also [13]). The optimal routing problem belongs to the class of multicommodity flow problems. The development of algorithms and software for optimal routing is an area of active research. Schwartz and Cheung ([20]) and Bertsekas ([3]) have studied, respectively, gradient and Newton projected primal methods, Fratta et al. ([10]) and Bertsekas and Gallager ([4]) have proposed flow deviation methods. Rockafellar ([19]) and Stern ([21]) have presented essentially dual methods. Authie ([1]) has studied a primal dual method. Distributed or parallel algorithms have also been proposed to solve multicommodity network flow problems (see [4], [7], [22] and [23]).

In this paper we propose a new approach for solving optimal routing problems in packet-switched networks. The method developed here is designed to obtain good rate of convergence while maintaining algorithmic simplicity and making effective use of parallel computing facilities. Most of the methods that can be found in the literature have the property that they allow multicommodity network flow problems to be decomposed into a set of smaller optimization problems at each major iteration (see [21], [19], [4], and [7]). These smaller problems correspond to single-commodity network flow problems. One of the main features of the method presented in this paper is to deal simultaneously with all commodities. The method presented here is a parallel dual method which takes advantage of the fact that the Lagrangian function can be decomposed by arcs. Minimization of the elementary Lagrangians is made by a quasi-Newton method and the dual functional is maximized by means of a modified approximate Newton method. The decomposition of the Lagrangian and the use of an approximate Newton method based on a block iterative algorithm render the dual method well suited for implementation on parallel computers or distributed systems.

The formulation of the problem is given in Section 2. Section 3 presents the dual method. Section 4 deals with the parallel dual method. A computational experience on a Transputer based distributed memory multiprocessor T-node 16-32 is presented in Section 5.

2. Problem formulation

Consider a directed graph with n nodes and a arcs. The graph incidence matrix is denoted by A. Let $D = \{d_1, ..., d_c\}$ be the set of destination nodes for network traffic. Let b_i^k be the average traffic input or output at node i associated with commodity k. Let f_j^k be the flow on arc j destined for d_k and $F_j = \sum_{k=1}^{c} f_j^k$ the total flow on arc j. Throughout the paper we adopt the following notational conventions:

$b^k = (b_1^k, ..., b_n^k)^t$, $f^k = (f_1^k, ..., f_a^k)^t$, $f = ({f^1}^t, ..., {f^c}^t)^t, f_j = (f_j^1, ..., f_j^c)^t$, $F = (F_1, ..., F_a)^t$.

The optimal routing problem can be stated as:

$$\min_F \sum_{j=1}^{a} g_j(F_j), \tag{1}$$

$$\text{subject to } Af^k - b^k = 0,\ k = 1, ..., c,$$

$$\text{and } f_j^k \geq 0,\ j = 1, ..., a,\ k = 1, ..., c.$$

In the case most commonly used in the packet-switched network literature (see [13] and [21]) we have:

for $j = 1, ..., a, g_j(F_j) = \left[\frac{1}{C_j - F_j} + T_j\right].F_j$, if $F_j \leq C_j$ and $f_j^k \geq 0$, $k = 1, ..., c$, and $g_j(F_j) = +\infty$ elsewhere.

The criterion is proportional to the message delay in the network averaged over all messages. This delay is computed under Kleinrock's ([12]) assumptions of independent Poisson arrival statistics and exponential message length distributions at each node. Moreover, for simplicity, it is assumed here that messages for all source-destination pairs have the same distribution with average message length equal to one bit per message. The term $\frac{1}{C_j - F_j}$ represents the average queuing and processing delay in the buffer. T_j is the propagation time along link j and C_j is the link capacity. In this paper we consider the following modified form of the cost function g_j :

$$g_j(f_j) = \left[\frac{1}{C_j - F_j} + T_j\right].F_j + r.\sum_{k=1}^{c}(\frac{1}{f_j^k}) + r'.\sum_{k=1}^{c}(f_j^k)^2,\ r > 0, r' > 0.$$

With the addition of quadratic and inverse terms g_j is converted to a strictly convex and continuously differentiable function of f_j. We note also that g_j is twice continuously differentiable. With r and r' sufficiently small the additional terms will not significantly alter the solution of problem (1). We remark that problem (1) is partially separable.

A dual problem is given by:

$$\max_{p \in R^{n \times c}} q(p), \tag{2}$$

where $q : R^{n \times c} \to R$ is the dual functional given by

$$q(p) = \min_f (\sum_{j=1}^{a} g_j(f_j) - \sum_{k=1}^{c} p^k.(Af^k - b^k)), \tag{3}$$

where $p^k = (p_1^k, ..., p_n^k)$ is referred to as the (row) vector of Lagrange multipliers associated with conservation of flow constraints relative to commodity k. The vector $p = (p^1, ..., p^c)$, is also referred to as a price vector and its components p_i^k as prices.

Adding the same constant to all coordinates of vector p^k leaves the dual cost unaffected. We can remove these c degrees of freedom by constraining the price of c nodes. Problem (2) is then strictly concave, twice continuously differentiable and subject to no constraints on the vector p (see [18, Section 26]). We have chosen to constrain prices p_n^k, $k = 1, ..., c$, to be zero.

The dual of a nonlinear programming problem is generally not easier to solve than the primal problem. However, for the optimal routing problem, which is partially separable, a dual method presents many advantages as we will see in the next two sections (see also the recent papers of Buckers [6] and Lootsma [14]).

3. A dual method

We first reorder the components of vectors f, p and b in order to facilitate the computations in the optimization process. Components of f, p and b are regrouped by arcs. Hence the dual functional can also be written:

$$q(p') = \min_{f'}(\sum_{j=1}^{a}(g_j(f_j) - p'.B(j).f_j)) + p'.b', \quad (4)$$

where $f' = (f_1^t, ..., f_a^t)^t$, $p' = (p_1, ..., p_{n-1})$, with $p_i = (p_i^1, ..., p_i^c)$, $b' = (b_1^t, ..., b_{n-1}^t)^t$, with $b_i = (b_i^1, ..., b_i^c)^t$, and $B(j) = \begin{pmatrix} B_1(j) \\ ... \\ B_{n-1}(j) \end{pmatrix}$ is a $((n-1).c \times c)$ matrix with blocks $B_i(j) = I$, the $(c \times c)$ identity matrix, if arc j is directed outbound from node i, $B_i(j) = -I$, if arc j is directed inbound to i, $B_i(j) = 0$, the $(c \times c)$ null matrix if arc j is not incident to node i ($(.)^t$ denotes transpose).

We use a modified approximate Newton algorithm in order to solve the dual problem (see in particular [11], [15, p. 281], and [9]). However, let us consider first the minimization of the Lagrangian. We note that the Lagrangian presents the good property to be decomposable into elementary Lagrangians which are all relative to a particular arc. We have chosen to minimize elementary Lagrangians by means of the Broyden Fletcher Goldfarb Shanno (BFGS) quasi-Newton method (for a complete study of quasi-Newton methods, reference is made to [8]). The BFGS method gives $\hat{f}'(p')$, the unique value of f' which minimizes the strictly convex Lagrangian, it gives also a symmetric, positive definite approximation, H, of the inverse of the Hessian, with respect to f', of the Lagrangian at point $\hat{f}'(p')$.

Let us consider now the solution of the dual problem by means of an approximate Newton algorithm.

The iterative algorithm starts at an arbitrary point p^0.

The vector flow, $\hat{f}'(p')$, which minimizes the Lagrangian, is obtained by means of the BFGS method. Hence, we can compute the gradient of the dual functional, which is given by:

$$\nabla_{p'} q(p') = (-B.\hat{f}'(p') + b')^t,$$

where B is the $((n-1).c \times a.c)$ matrix with blocks $B_i(j)$.

We can also compute a symmetric, negative definite approximation, Q, of the Hessian of the dual functional, where Q is given by: $Q = -B.H.B^t$ (see [17] and [15, p. 281]).

We note that the computation of Q is very easy since the blocks $B_i(j)$ of matrix B are identity, minus identity, or null matrices. Matrix Q is represented on figure 1 (H_j denotes the inverse of the Hessian of the elementary Lagrangian relative to arc j).

Then, we can solve approximately the system:

$$d.Q = -\nabla_{p'} q(p'), \tag{6}$$

by an underrelaxed block Jacobi algorithm and we can compute a new price vector:

$$p' + d.$$

We note that the underrelaxed block Jacobi algorithm iterates on vector d according to directions which minimize the cost $-\frac{1}{2}.d.Q.d^t - \nabla_{p'} q(p').d^t$. If the block Jacobi algorithm is initialized with $d = 0$ and if the relaxation parameter is sufficiently small, it will converge (see [5, p. 154]) and we will have:

$$-\frac{1}{2}.d^k.Q.d^{k^t} - \nabla_{p'} q(p').d^{k^t} < 0, \forall k.$$

and since $-Q$ is positive definite we have:

$$\nabla_{p'} q(p').d^{k^t} > 0, \forall k.$$

and d^k is an ascent direction whatever k (see [5, p. 202]).

This approximate Newton algorithm will be referred to in the following as algorithm 1.

If we compute only the diagonal blocks of the approximation matrix Q we obtain a second approximation matrix, Q' of the Hessian matrix and a second algorithm referred to in the following as algorithm 2. In this case the linear system:

$$d.Q' = -\nabla_{p'} q(p'),$$

is constituted by n independents subsystems and we can apply directly the Cholesky method instead of a block Jacobi algorithm.

4. Parallel Dual Method

The decomposition of the Lagrangian and the use of an approximate Newton method based on a block-Jacobi algorithm (for the solution of the dual problem) allow implementation of the dual method on parallel computers. A degree of parallelism can be introduced in both the minimization of the Lagrangian and the linear algebra of the approximate Newton method. The algorithm is partitioned into tasks that can be executed by different processing units. We have chosen a partition of the algorithm that tends to minimize data communications between processors. This partition results from the decomposition of the network into subnetworks, each subnetwork being associated with a processor.

Each subnetwork contains a set of nodes called main nodes. Any two different subnetworks do not share any main node. Each subnetwork contains also a set of arcs called main arcs. A main arc connects two main nodes of the same subnetwork. Arcs between two main nodes of different subnetworks are called border arcs. A border arc is assigned to one and only one subnetwork. Auxiliary nodes are created in order to assign an origin or a destination to a border arc. We note that each subnetwork is connected. For each network topology, we look for a decomposition of the network which balances the number of main nodes and the number of arcs in each subnetwork and which minimizes and balances the number of main nodes of other subnetworks which are connected to the main nodes of each subnetwork. This strategy will tend to balance the computation and communication loads. Figure 2 shows a mesh network with 16 nodes and 24 arcs. An example of decomposition of the network of figure 2 is given in figure 3, where auxiliary nodes and border arcs are represented, respectively, by thick cycles, and arrows.

Each processor implements the dual method presented in Section 3 on its own subnetwork. Clearly the minimizations of the elementary Lagrangians can be made independently. However, the maximization of the dual functional requires data communication between the different processors and synchronization of the processors. The data transferred between processors are relative to flows which minimize elementary Lagrangians, inverse Hessians of elementary Lagrangians and prices.

Concerning the computation of the gradient of the dual functional, data communications are similar for algorithms 1 and 2. Each processor sends partial calculus of the gradient component relative to an auxiliary node to the neighbor which possesses this node as a main node. This neighbor sends, in turn, complete calculus of the gradient component relative to this node to all neighbors which possess this node as an auxiliary node. The complete calculus consists of the sum of all partial calculus relative to a particular node.

As concerns the computation of an approximation of the Hessian of the dual functional, data communications are different for algorithms 1 and 2. For algorithm 1, each processor T_i, sends partial calculus of the block line, Q^i_j, of matrix Q, relative to an auxiliary node j, to its neighbor which possesses node j as a main node. This neighbor, in turn, computes the complete block line of Q, relative to node j, by making the sum of all bloc lines relative to j. Figure 4 shows (for the decomposition shown in figure 3) the block lines of each processor and data communications. For algorithm 2, block lines are replaced by blocks.

We consider now the connected network obtained by joining up again each subnetwork without auxiliary nodes. This network is similar to the original network and it is constituted by main nodes, main arcs and border arcs. Concerning the approximate solution of linear system of equations (6), for algorithm 1, each processor, T_i, sends to its neighbors, T_j, at each iteration k, the value d_l^k relative to main nodes l of T_i which are connected to a node of T_j by a border arc. When the iteration process terminates, each processor computes the prices of its main nodes and the prices of the main nodes of other processors which are connected to its main nodes by a border arc.

We recall that algorithm 2 uses a direct method for the solution of the linear system of equations. However, in order to minimize elementary Lagrangians relative to border arcs, each processor needs to receive from its neighbors the prices of its auxiliary nodes.

5. Computational Experience

Experiments with the dual method were carried out on a T-node 16-32 multiprocessor, which is a Transputer based distributed memory machine. Table 1 gives the numbers of iterations, computational times, speedups, and efficiencies obtained with 1, 2, 4, and 8 processors for a mesh network with 16 nodes, 24 arcs, and 3 commodities. Table 2, 3 and 4 gives, respectively,the corresponding results for mesh networks with 16 nodes, 24 arcs, and 5 commodities, 38 nodes, 66 arcs, and 3 commodities, and 48 nodes, 82 arcs, and 3 commodities. In particular network topology for problem 1 and 2 is given by figure 2. For each problem we have tried to balance the number of main nodes and the number of arcs in each subnetwork and to minimize the number of main nodes of other subnetwoks connected to the main nodes of each subnetwork. The network topologies of the distributed memory multiprocessor for 4 and 8 processors are also mesh network topologies.

Tables 1, 2, 3, and 4 show that algorithm 2 is faster than algorithm 1. Clearly, algorithm 2 needs to approximate only the diagonal blocks of the Hessian of the dual functional. Moreover, in the case of algorithm 2, equation (6) is equivalent to $n-1$ independent systems of c equations, which is more easy to solve than a system of $(n-1).c$ equations, as it is the case for algorithm 1.

We note that parallel implementation speeds up efficiently algorithms 1 and 2. The efficiencies are close to 0.65. Other computational experiences ([16] and [7]) show that parallel algorithms for nonlinear optimization problems have in general an efficiency close to 0.65. This is mainly due to difficulties of load balancing for iterative nonlinear algorithms. The efficiency decreases as the number of processors increases, because the number of communications and synchronisations increases.

The speedups are in general better for algorithm 2 than for algorithm 1 because algorithm 1 needs a higher amount of synchronization and communication than algorithm 2. Exceptions may occur because of the difficulties of load balancing, in particular for the minimization of elementary Lagrangians.

We note also that algorithm 2 needs more iterations than algorithm 1 in order to converge. We see that the number of iterations varies with the number of processors. This is due to the fact that the order in which the components of vector d are reactualized varies with the number of processors.

References

[1] G. Authie, Contribution à l'optimisation de flots dans les réseaux. Un multiprocesseur expérimental pour l'étude des itérations asynchrones, Thèse de Doctorat d'Etat, UPS Toulouse, 1987.

[2] D. P. Bertsekas and D. El Baz, Distributed asynchronous relaxation methods for convex network flow problems, *SIAM Journal on Control and Optimization*, 25 (1987), 74-85.

[3] D. P. Bertsekas and M. Gafni, Projected Newton methods and optimization of multicommodity flows, *IEEE Trans. Automat. Control*, 28, (1983), 1090-1096.

[4] D. P. Bertsekas and R. Gallager, *Data Networks* (Prentice Hall, Englewood Cliffs, N. J., 1987).

[5] D. P. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation*, (Prentice Hall, Englewood Cliffs, N. J., 1989).

[6] R. Buckers, Numerical experiments with dual algorithm for partially separable nonlinear optimization problems, in: D. Evans et al. eds., *Proceedings of the Parallel Computing 89 Conference* (Elsevier Science Publishing, North Holland, 1990), 555-562.

[7] R. Chen and R. Meyer, Parallel optimization for traffic assignment,*Mathematical Programming, Series B*, 42 (1988), 327-346.

[8] J. Dennis and J. Moré, Quasi-Newton methods, motivation and theory, *SIAM Review,* 19 (1977), 46-88.

[9] R. Dembo and T. Steihaug, Truncated-Newton algorithms for large-scale unconstrained optimization, *Mathematical Programming* , 26 (1983), 190-212.

[10] L. Fratta, M. Gerla and L. Kleinrock, The flow deviation method: an approach to store-and-forward communication network design, , *Networks* , 3, (1973), 97-133.

[11] R. Fletcher, Methods related to Lagrangian functions, in: Gill and Murray eds., *Numerical Methods for Constrained Optimization* (Academic Press, London, 1974), 219-239.

[12] L. Kleinrock, *Communication Nets: Stochastic Message Flow and Delay* (Mc Graw-Hill, New York, 1964).

[13] L. Kleinrock, *Queuing Systems* (John Wiley, New York, 1976).

[14] F. Lootsma, Exploitation of structure in nonlinear optimization, in: D. J. Evans et al. editors, *Proceedings of the Parallel computing 89 Conference,* (Elsevier Science Publishing B. V. North Holland, 1990), 31-45.

[15] M. Minoux, *Programmation Mathématique,* (Dunod, Paris, 1983).

[16] S. Nash and A. Sofer, Block truncated Newton methods for parallel optimization, *Mathematical Programming,* 45 (1989), 529-546.

[17] C. Ribeiro and D. El Baz, A dual method for optimal routing in packet switched networks, IFIP Conference on System Modelling and Optimization, Zurich (September 2-6, 1989).

[18] R. Rockafellar, *Convex Analysis,* (Princeton University Press, Princeton, 1970).

[19] R. Rockafellar, *Network Flows and Monotropic Optimization,* (John Wiley & Sons, New York, 1984).

[20] M. Schwartz and C. Cheung, The gradient projection algorithm for multiple routing in message-switched networks, *Proc. Fourth annual Data Communications Symposium*, Oct. 7-9 1975, Quebec city, Canada.

[21] T. Stern, A class of decentralized routing algorithms using relaxation, *IEEE Trans. on Communications*, COM 25 (1977), 1092-1102.

[22] K. Tsai, G. Huang, K. Antonio, T. Tsai, Distributed iterative aggregation algorithms for box constrained minimization problems and optimal routing in data networks, *IEEE Trans. on Automat. Control,* 34 (1989), 34-46.

[23] J. Tsitsiklis and D. P. Bertsekas, Distributed asynchronous optimal routing in data networks, *IEEE Trans. on Automat. Control,* 31 (1986), 325-332.

$$Q = - \begin{array}{|c|c|c|c|} \hline B_1(1) . H_1 . B_1^T(1) + \dots + B_1(a) . H_a . B_1^T(a) & B_1(1) . H_1 . B_2^T(1) + \dots + B_1(a) . H_a . B_2^T(a) & \dots & B_1(1) . H_1 . B^T_{n-1}(1) + \dots + B_1(a) . H_a . B^T_{n-1}(a) \\ \hline \vdots & & & \\ \hline B_{n-1}(1) . H_1 . B_1^T(1) + \dots + B_{n-1}(a) . H_a . B_1^T(a) & & \dots & B_{n-1}(1) . H_1 . B^T_{n-1}(1) + \dots + B_{n-1}(a) . H_a . B^T_{n-1}(a) \\ \hline \end{array}$$

Figure 1: Hessian dual matrix

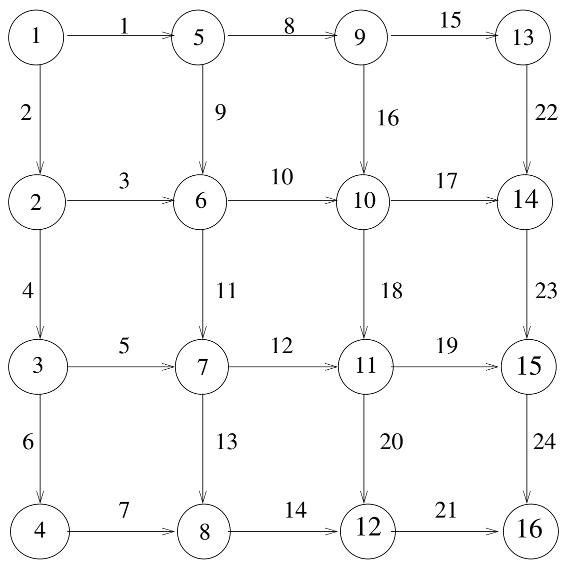

Figure 2: network topology

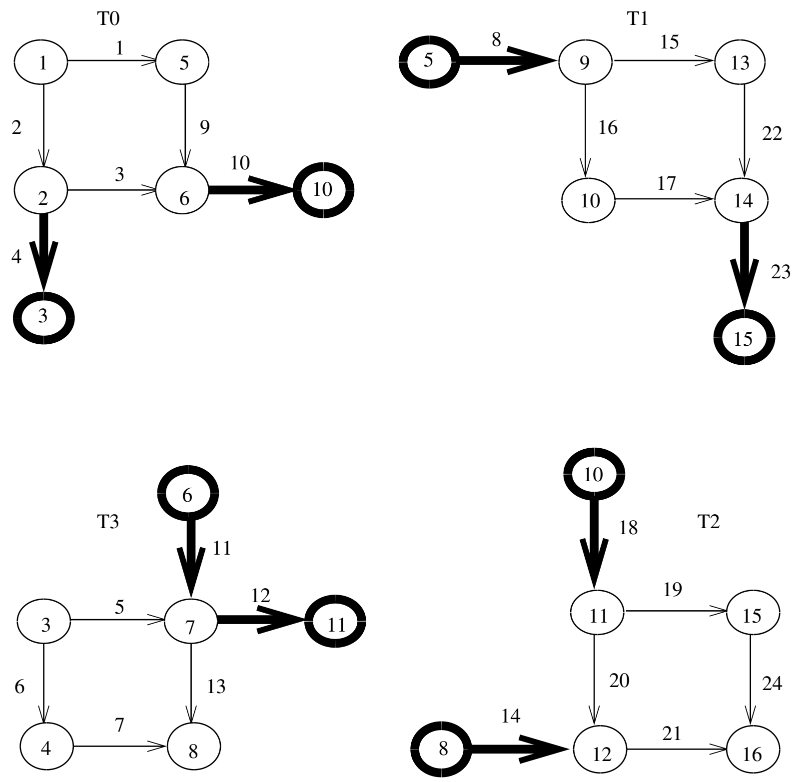

Figure 3: partition of the network for 4 processors

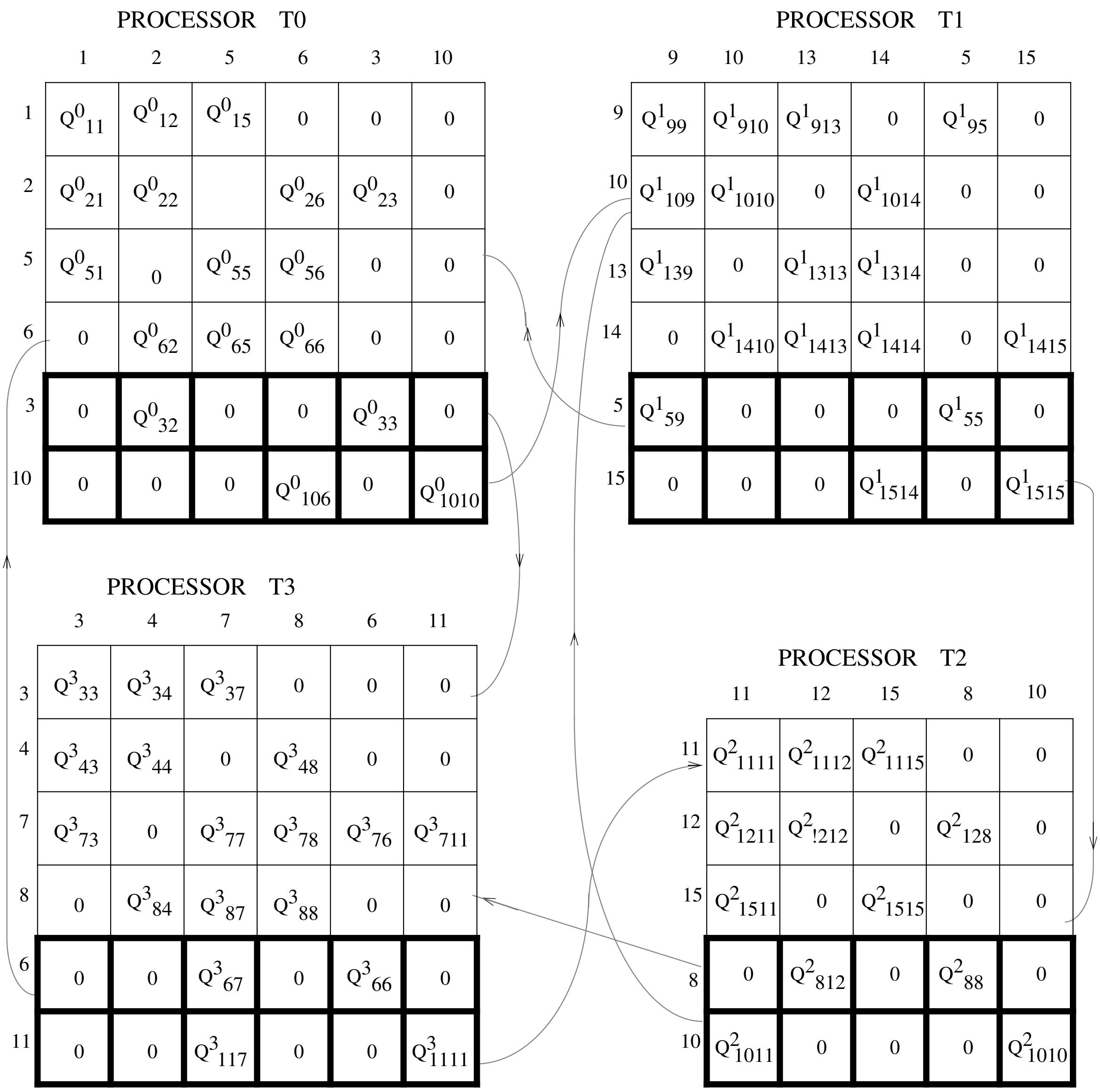

Figure 4: partial block lines of the dual Hessian and data communications

Number of processors		1	2	4	8
Alg. I	objective value	18.289	18.291	18.29	18.291
	time (sec)	566.10	355.24	199.84	112.64
	iterations	693	669	685	687
	speed-up	----	1.59	2.83	5.03
	efficiency	1.0	0.79	0.70	0.63
Alg. II	objective value	18.29	18.29	18.29	18.29
	time (sec)	356.7	217.47	131.79	77.09
	iterations	715	724	726	726
	speed-up	----	1.64	2.71	4.63
	efficiency	1.0	0.82	0.68	0.58

Table 1: computational results for problem 1.

Number of processors		1	2	4	8
Alg. I	objective value	27.497	27.497	27.497	27.497
	time (sec)	1250.94	842.13	443.697	267.17
	iterations	835	870	852	857
	speed-up	----	1.49	2.82	4.68
	efficiency	1.0	0.75	0.70	0.58
Alg. II	objective value	27.496	27.496	27.498	27.49
	time (sec)	905.86	577.58	321.06	205.57
	iterations	896	1010	981	1010
	speed-up	----	1.57	2.82	4.41
	efficiency	1.0	0.78	0.70	0.55

Table 2: computational results for problem 2.

Number of processors		1	2	4	8
Alg. I	objective value	29.521	29.522	29.522	29.522
	time (sec)	6857.88	5064.75	2736.31	1465.81
	iterations	1305	1406	1404	1346
	speed-up	----	1.35	2.5	4.68
	efficiency	1.0	0.675	0.625	0.585
Alg. II	objective value	29.525	29.526	29.526	29.526
	time (sec)	2427.17	1700.68	849.87	437.85
	iterations	2177	2250	2209	1957
	speed-up	----	1.43	2.86	5.54
	efficiency	1.0	0.71	0.71	0.69

Table 3: computational results for problem 3.

Number of processors		1	2	4	8
Alg. I	objective value	35.71	35.71	35.71	35.71
	time (sec)	11275.3	8445.9	4517.5	2315.26
	iterations	1268	1410	1488	1319
	speed-up	----	1.34	2.5	4.87
	efficiency	1.0	0.67	0.62	0.61
Alg II	objective value	35.72	35.72	35.72	35.72
	time (sec)	2995.5	1833.75	1057.89	574.0
	iterations	2230	2169	2274	2238
	speed-up	----	1.63	2.83	5.22
	efficiency	1.0	0.81	0.71	0.65

Table 4: computational results for problem 4.