Paper

Distributed, asynchronous algorithms for network control with contracted flow rates – a review

Andrzej Karbowski

Abstract— The paper reviews current algorithms for distributed, asynchronous control of networks when the customer is guaranteed to get some predetermined (e.g., as a part of a service level agreement – SLA) values of flow. Two cases are considered – both with single and multiple commodity. It is assumed, that the flow cost functions are convex with special attention devoted to linear and strictly convex cases.

Keywords— computer networks, asynchronous algorithms, distributed optimization, routing, quality of service, service level agreement.

1. Introduction

In the network optimization (e.g., in the Internet traffic control) one of the most important problems is elaboration of such policies, which guarantee that the service provider will deliver to the customer the contracted amount of the commodity flow (bandwidth in data networks). This amount is rigid and usually specified in service level agreement (SLA).
One may distinguish two situations:

- When the operator is obliged to deliver to all network nodes (customers) the contracted amount of flow of the same commodity, no matter from where; such situation is typical for power and water networks; in the Internet subnetworks with contracted access rates and many connection points to other operators' subnetworks as well as some peer-to-peer (P2P) computational grids may be modelled in this way.
- When the operator is responsible for the delivery of different commodities at the contracted level. Such problems are characteristic for transportation networks, and for data networks with higher quality of service (QoS) demands, such as networks with virtual connections (that is with isochronous traffic), for example: voice over Internet Protocol (VoIP), video-conferences, video on demand, etc.

Because networks are bigger and bigger, it is more and more difficult to control them effectively. Hence, in the recent years a special attention is paid to decentralized optimization and control algorithms. Especially, their asynchronous versions catch great attention of scientists dealing with management and control of networks. It is due to the flexibility and scalability of asynchronous algorithms. In such algorithms, the computing nodes may use outdated information and communicate with each other at different, random times, but this does not destroy the convergence. Moreover, the situation in networks, for example traffic in data networks, is changing dynamically. Asynchronous algorithms, using always the latest information, which is measured in the neighbourhood of the computing nodes, lead the network towards the current global optimum. Nowadays, for such a big and nonhomogenous network as Internet, it is even difficult to imagine an effective optimization mechanism that would not be distributed and asynchronous!

In the article such algorithms for single and multicommodity networks with contracted flow rates will be presented. Several different approaches will be considered, such as: standard primal-dual, ε-relaxation, an approach based on minimum first derivative length principle, an approach based on multiple adaptive traffic engineering method. It is assumed, that the flow cost functions are convex with special attention devoted to linear and strictly convex cases.

This paper is complementary to the paper "Distributed asynchronous algorithms in the Internet – new routing and traffic control methods" [7] presented at *DSTIS 2004 Conference* and closes the series of review papers devoted to distributed asynchronous network algorithms presented by the author at DSTIS conferences.

2. The optimization of flows in single commodity networks with linear and nonlinear cost functions

2.1. General convex functions

We consider a directed graph consisting of n nodes (routers). Let us denote by N the set of all these nodes, by A the set of all arcs (that is, the set of all links in the network) and by N_i the set of neighbours of the ith node (that is, the set of all nodes from the set N, to which arcs starting from i go). Let us assume, that every arc $(i,j) \in A$ is characterized by a continuous, convex function $a_{ij}(f_{ij})$ of the cost of the realization of the flow f_{ij} from the node i to j and box flow constraints:

$$f_{ij} \in F_{ij} = [b_{ij}, c_{ij}]. \tag{1}$$

With every node $i \in N$ we connect a given supply $s_i > 0$ or demand $s_i < 0$. Our goal is the calculation of such

a distribution of flows between nodes, that all demands are satisfied and the total cost of these flows is minimal, that is

$$\min_{f} \sum_{(i,j)\in A} a_{ij}(f_{ij}), \tag{2}$$

$$\sum_{\{j|(j,i)\in A\}} f_{ji} + s_i = \sum_{\{j|(i,j)\in A\}} f_{ij}, \ \forall i \in N, \tag{3}$$

$$b_{ij} \le f_{ij} \le c_{ij}, \ \forall (i,j) \in A, \tag{4}$$

where f is a vector of all flows. Equations (3) result from the balance of flows in the nodes (so-called 1st Kirchhoff rule).

We assume, that supplies and demands s_i are balanced over the network, that is:

$$\sum_{i\in N} s_i = 0 \tag{5}$$

and, of course, that the set of admissible solutions (i.e., the set of possible combinations of coordinates of the vector of flows f) is not empty.

In the Internet such problem may be important for wide area operators, who have several connection points with other operators.

To solve this problem we formulate a Lagrange function [1], but taking into account only balance constraints (3). If we denote the multiplier corresponding to the ith node equation as p_i, it will be:

$$L(f,p) = \sum_{(i,j)\in A} a_{ij}(f_{ij}) + \sum_{i\in N} p_i \left(\sum_{\{j|(j,i)\in A\}} f_{ji} + s_i - \sum_{\{j|(i,j)\in A\}} f_{ij} \right). \tag{6}$$

Making some simple transformations we can present the Lagrange function in the following way:

$$L(f,p) = \sum_{(i,j)\in A} \Big[a_{ij}(f_{ij}) - (p_i - p_j) f_{ij} \Big] + \sum_{i\in N} p_i s_i. \tag{7}$$

Hence, the dual function in problem (2)–(4) will have the form:

$$\begin{aligned} L_D(p) &= \min_{b\le f\le c} \left\{ \sum_{(i,j)\in A} \Big[a_{ij}(f_{ij}) - (p_i - p_j) f_{ij} \Big] + \sum_{i\in N} p_i s_i \right\} \\ &= \sum_{(i,j)\in A} \left\{ \min_{b_{ij}\le f_{ij}\le c_{ij}} \Big[a_{ij}(f_{ij}) - (p_i - p_j) f_{ij} \Big] \right\} + \sum_{i\in N} p_i s_i \\ &= \sum_{(i,j)\in A} L_{ij}(p_i - p_j) + \sum_{i\in N} p_i s_i, \end{aligned} \tag{8}$$

where L_{ij} is a component of the dual function corresponding to the arc (i,j), that is:

$$L_{ij}(p_i - p_j) = \min_{b_{ij}\le f_{ij}\le c_{ij}} \Big[a_{ij}(f_{ij}) - (p_i - p_j) f_{ij} \Big]. \tag{9}$$

According to the duality theory [1, 6], the solution of the problem (2)–(4) may be obtained by the solution of the dual problem:

$$\max_{p\in\mathbb{R}^n} L_D(p). \tag{10}$$

To find the optimal solution of the problem (10) one may use gradient of the dual function L_D, with coordinates:

$$\begin{aligned} \frac{\partial L_D}{\partial p_i} &= -\sum_{\{j|(j,i)\in A\}} L'_{ji}(p_j - p_i) + \sum_{\{j|(i,j)\in A\}} L'_{ij}(p_i - p_j) + s_i \\ &= \sum_{\{j|(j,i)\in A\}} f_{ji} - \sum_{\{j|(i,j)\in A\}} f_{ij} + s_i. \end{aligned} \tag{11}$$

The equivalent statement of optimality conditions stemming from duality theory is, that a flow vector $\hat{f}$ is optimal if and only if it is primal feasible, that is $\hat{f}_{ij} \in F_{ij} \ \forall \ i,j$ and there exists a price vector $\hat{p}$ satisfying together with $\hat{f}$ the following conditions (called complementary slackness conditions – CS) [3]:

$$a^-_{ij}(\hat{f}_{ij}) \le \hat{p}_i - \hat{p}_j \le a^+_{ij}(\hat{f}_{ij}). \tag{12}$$

In this expression leftmost and rightmost are, respectively, the left and the right derivatives of the arc cost function. Usually we deal with smooth cost functions and we have:

$$a^-_{ij}(\hat{f}_{ij}) = a^+_{ij}(\hat{f}_{ij}) = a'_{ij}(\hat{f}_{ij}). \tag{13}$$

In practical numerical calculations, while looking for a good approximation of the optimal solution, a relaxed version of the CS conditions proved to be very useful. For a given scalar $\varepsilon > 0$ inequalities (12) are replaced by the following:

$$a^-_{ij}(f_{ij}) - \varepsilon \le p_i - p_j \le a^+_{ij}(f_{ij}) + \varepsilon. \tag{14}$$

These conditions are called ε-complementary slackness conditions, ε-CS for short. The optimization approach which applies ε-CS conditions is called ε-relaxation method [3]. It consists in adjusting flows ("flow push") and increasing prices ("price rise") at appropriate nodes in such a way, that ε-CS conditions are maintained. This algorithm may be implemented in a distributed, asynchronous version [2], where each node i is a processor that updates its own price and its arcs flows, and exchanges information with its forward

$$F_i = \{j|(i,j)\in A\} \tag{15}$$

and backward

$$B_i = \{j|(j,i)\in A\} \tag{16}$$

adjacent nodes.

The information available at node i for any time t is as follows:

- $p_i(t)$: the price of node i;
- $p_j(i,t)$: the price of node $j \in F_i \cup B_i$ communicated by j to i at some earlier time;
- $f_{ij}(i,t)$: the estimate of the flow on the arc $(i,j), j \in F_i$, available at node i at time t;
- $f_{ji}(i,t)$: the estimate of the flow of arc $(j,i), j \in B_i$ available at node i at time t.

At each time t, each node i may be in one of the following four phases:

1. *Idle phase.* Node i does nothing.

2. *Computational phase.* Node i computes the surplus $g_i(t)$:

$$g_i(t) = \sum_{j \in B_i} f_{ji}(i,t) - \sum_{j \in F_i} f_{ij}(i,t) + s_i \,. \tag{17}$$

If $g_i(t) < 0$, node i does further nothing. Otherwise the following values:

$$p_i(t),\ \ f_{ij}(i,t),\ j \in F_i,\ \ f_{ji}(i,t),\ j \in B_i \tag{18}$$

are updated. The updating is performed due to the following procedure:

Step 1: (Calculation of the push list and the flow margin)
Given a flow-price vector satisfying the ε-CS conditions, *the push list* L_i of node $i, \forall i \in N$, is defined as follows:

$$\begin{aligned} L_i \ = \ & \{(i,j) | \varepsilon/2 < p_i(t) - p_j(i,t) \\ & - a_{ij}^{+}(f_{ij}(i,t)) \le \varepsilon\} \\ & \cup \{(j,i) | -\varepsilon \le p_j(i,t) - p_i(t) \\ & - a_{ji}^{-}(f_{ji}(i,t)) < -\varepsilon/2\} \,. \end{aligned} \tag{19}$$

For each arc (i,j) or (j,i) in the push list L_i, the supremum of σ for which

$$p_i(t) - p_j(i,t) \ge a_{ij}^{+}(f_{ij}(i,t) + \sigma) \tag{20}$$

or, respectively,

$$p_j(i,t) - p_i(t) \le a_{ji}^{-}(f_{ij}(i,t) - \sigma) \,,$$

is called the *flow margin*.

Step 2: (Scan of the push list)
If $L_i = \emptyset$ go to Step 4.

Step 3: (δ-Flow push)
Choose an arc from the push list L_i and let

$$\delta = \min(g_i(t), \text{ flow margin of the chosen arc}). \tag{21}$$

Increase f_{ij} by δ if (i,j) is the arc, or decrease f_{ji} by δ if (j,i) is the arc. If as a result the surplus becomes zero, go to the next iteration; otherwise, go to Step 2.

Step 4: (Price rise)
Increase the price p_i by the maximum amount that maintains ε-CS conditions. Go to the next iteration.

3. *Output phase.* The values of $p_i(t), f_{ij}(i,t), f_{ji}(i,t)$, computed during the computational phase, are communicated to the adjacent nodes $j \in F_i \cup B_i$.

4. *Input phase.* Node i receives from one or more adjacent nodes $j \in F_i \cup B_i$ a message containing the price $p_j(t')$ and the arc flow $f_{ij}(j,t')$ (when $j \in F_i$) or $f_{ji}(j,t')$ (when $j \in B_i$), computed by node $j, j \in F_i \cup B_i$, at some earlier time $t' < t$.
On the basis of this information, the node i updates $p_j(i,t)$ and $f_{ij}(i,t)$ if $j \in F_i$, ($f_{ji}(i,t)$, if $j \in B_i$).
If $p_j(t') \ge p_j(i,t)$, then $p_j(i,t) = p_j(t')$.
In addition, if $j \in F_i$, the value of $f_{ij}(i,t)$ is replaced by $f_{ij}(j,t')$ if

$$p_i(t) < p_j(t') + a_{ij}^{+}(f_{ij}(j,t')) + \varepsilon \ \text{ and } \ f_{ij}(j,t') < f_{ij}(i,t). \tag{22}$$

In the case of $j \in B_i$, the value of $f_{ji}(i,t)$ is replaced by $f_{ji}(j,t')$ if

$$p_j(t') \ge p_i(t) + a_{ji}^{-}(f_{ji}(j,t')) - \varepsilon \ \text{ and } \ f_{ji}(j,t') > f_{ji}(i,t). \tag{23}$$

The algorithm terminates if there is a time t_k such that, for all $t \ge t_k$:

$$g_i(t) = 0 \ \ \forall i \in N \,,$$

$$f_{ij}(i,t) = f_{ij}(j,t) \ \ \forall (i,j) \in A \,,$$

$$p_j(t) = p_j(i,t) \ \ \forall j \in F_i \cup B_i \,.$$

It may be shown, that the algorithm converges if the initial prices and flows satisfy ε-CS conditions, the nodes never stop executing iterations and communication and assuring that the old information is eventually purged from the system [2].

2.2. Linear cost functions

In this case the optimization problem has the following form:

$$\min_{f} \sum_{(i,j) \in A} a_{ij}(f_{ij}) = \alpha_{ij} f_{ij} \,, \tag{24}$$

$$\sum_{\{j | (j,i) \in A\}} f_{ji} + s_i = \sum_{\{j | (i,j) \in A\}} f_{ij}, \ \ \forall i \in N \,, \tag{25}$$

$$b_{ij} \le f_{ij} \le c_{ij}, \ \forall (i,j) \in A \,. \tag{26}$$

We may apply the algorithm presented in Subsection 2.1 in a simplified version [4][1], taking as:

- arc cost derivatives:

$$a_{ij}^{'}(f_{ij}) = \alpha_{ij}, \tag{27}$$

[1] Chronologically the asynchronous version of the ε-relaxation algorithm for linear minimum cost flow problems was presented much earlier in [4]; the version for convex problems from [2] was its extension.

- the ε-CS conditions:

$$f_{ij} < c_{ij} \Rightarrow p_i - p_j \leq \alpha_{ij} + \varepsilon, \ \forall (i,j) \in A, \quad (28)$$

$$b_{ij} < f_{ij} \Rightarrow p_i - p_j \geq \alpha_{ij} - \varepsilon, \ \forall (i,j) \in A, \quad (29)$$

- the push list:

$$L_i = \{(i,j) | \, p_i(t) = p_j(i,t) + \alpha_{ij} + \varepsilon \ \text{ and } \ f_{ij}(i,t) < c_{ij}\}$$
$$\cup \{(j,i) | \, p_i(t) = p_j(i,t) - \alpha_{ji} + \varepsilon \ \text{ and } \ b_{ji} < f_{ji}(i,t)\}, \quad (30)$$

- the flow margin:

$$\sigma = \begin{cases} c_{ij} - f_{ij}(i,t) & j \in F_i \\ f_{ji}(i,t) - b_{ji} & j \in B_i \end{cases}, \quad (31)$$

- the replacement conditions for flow estimates ((22) and (23)):
  - in the case of $j \in F_i$, the value of $f_{ij}(i,t)$ is replaced by $f_{ij}(j,t')$ if

$$p_i(t) < p_j(t') + \alpha_{ij} \ \text{ and } \ f_{ij}(j,t') < f_{ij}(i,t), \quad (32)$$

  - in the case of $j \in B_i$, the value of $f_{ji}(i,t)$ is replaced by $f_{ji}(j,t')$ if

$$p_j(t') \geq p_i(t) + \alpha_{ji} \ \text{ and } \ f_{ji}(j,t') > f_{ji}(i,t). \quad (33)$$

Application to the shortest path problem. For a single connection it is also possible to formulate a shortest path problem to find a shortest path from node s to node d as the following linear minimum flow cost problem [3]:

$$\min_f \sum_{(i,j) \in A} \alpha_{ij} f_{ij}, \quad (34)$$

$$\sum_{\{j|(j,i) \in A\}} f_{ji} - \sum_{\{j|(i,j) \in A\}} f_{ij} = \begin{cases} -1 & \text{if } i = s \\ 1 & \text{if } i = d \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in N, \quad (35)$$

$$0 \leq f_{ij}, \ (i,j) \in A. \quad (36)$$

It may be solved in a distributed, asynchronous way through the ε-relaxation algorithm. The optimal path will be made of those arcs (i,j) for which $\hat{f}_{ij} = 1$ (for the remaining $\hat{f}_{ij} = 0$).

2.3. Strictly convex arc cost functions

In this section we assume, that the functions $a_{ij}(f_{ij}), (i,j) \in A$ in problem (2)–(5) are strictly convex. This problem, both the formulation and the basic features were taken from the book [4].

First, let us notice from Eqs. (7) and (5), that the optimal Lagrange multipliers are not unique, because one may add to all of them the same constant and the function value will not change. Hence, it is worthwhile to fix one of the coordinates of the vector p and take for example:

$$p_1 = r, \quad (37)$$

where r is an arbitrary nonzero real constant.

To solve the dual optimization problem (10) it is necessary to solve the family of scalar optimization problems (9), separately for every arc. They are very easy to solve, often even analytically.

According to the duality theory [1, 6] the function L_D and all functions L_{ij} are concave. It can be proved [4], that the algorithm of the Lagrange multiplier iteration of the form:

$$p_i := \begin{cases} r & i = 1 \\ \arg\max\limits_{\xi} L_D(p_1, p_2, \ldots, p_{i-1}, \xi, p_{i+1}, \ldots, p_n) & i = 2,3,\ldots,n \end{cases} \quad (38)$$

is also an order preserving mapping and it is convergent in a totally asynchronous version.

In fact, it is not necessary to perform optimization of L_D with the ith coordinate. To explain it let us define for every $i \in N \setminus \{1\}$ a *point-to-set* mapping R_i, which assigns to every Lagrange multipliers vector p a set of all prices which maximize the dual function L_D with respect to the ith price p_i, that is:

$$R_i(p) = \arg\max_{\xi} L_D(p_1, p_2, \ldots, p_{i-1}, \xi, p_{i+1}, \ldots, p_n). \quad (39)$$

It can be proved, that if the problem (2)–(4) and (5) is feasible, that is:

$$\sum_{\{j|(i,j) \in A\}} b_{ij} - \sum_{\{j|(j,i) \in A\}} c_{ji} \leq s_i \leq \sum_{\{j|(i,j) \in A\}} c_{ij} - \sum_{\{j|(j,i) \in A\}} b_{ji} \quad (40)$$

then the set $R_i(p)$ is either a singleton or a closed interval. Due to the concavity and differentiability of L_D every point $\xi \in \mathbb{R}$ belonging to $R_i(p)$ is a root of the scalar equation:

$$\frac{\partial L_D(p_1, p_2, \ldots, p_{i-1}, \xi, p_{i+1}, \ldots, p_n)}{\partial p_i} = 0 \quad (41)$$

that is, due to Eq. (11):

$$R_i(p) = \left\{ \xi : \sum_{\{j|(j,i) \in A\}} L'_{ji}(p_j - \xi) = \sum_{\{j|(i,j) \in A\}} L'_{ij}(\xi - p_j) + s_i \right\}. \quad (42)$$

Let us denote now as $\underline{R}_i(p)$ and $\overline{R}_i(p)$, respectively, the left and the right end of this interval. It turns out, that both these functions preserve the order. In the consequence,

the algorithm of the Lagrange multiplier iteration of the form:

$$p_i := \begin{cases} r & i = 1 \\ \gamma \underline{R}_i(p) + (1-\gamma)\overline{R}_i(p) & i = 2,3,\ldots,n \end{cases} \quad (43)$$

is equivalent to (38) for every $\gamma \in [0,1]$ and of course also convergent in a totally asynchronous version [4].
If it is not difficult to calculate the intervals $R_i(p), i = 1,\ldots,n$, one may propose another algorithm, which is convergent under the partial asynchronism assumptions [4, 9]. In this algorithm the ith coordinate is changed due to the iteration:

$$p_i := \gamma \cdot p_i + (1-\gamma) \arg \min_{\xi \in R_i(p)} |\xi - p_i| \;\; i = 1,2,3,\ldots,n \quad (44)$$

with $0 < \gamma < 1$. The computational experiments [9] showed, that this algorithm is considerably faster than the algorithm (43).

3. The optimization of flows in multicommodity networks with contracted transmission rates for virtual connections

Now we will consider a more complicated situation, where from the network we expect not only the transport of the total volume of traffic, from all sources to all destination nodes, but also the guarantees on the flow between given pairs of nodes. So we will deal with networks which actually provide virtual connections, in other words with virtual-circuit data networks.
We define a set W of origin-destination pairs and assume, that for every connection $w = (s,d)$, $s,d \in N, s \neq d$ the total flow r_w may be split to several paths P_w. We also assume, that the sets P_w for different w are disjoint. We will denote by A_p the set of all arcs (links) belonging to (i.e., forming) the path p.
Let x_p be the flow through a particular path $p \in P_w$. According to our assumptions, the flow f_{ij} through an arc (i,j) equals:

$$f_{ij} = \sum_{p \in P_{ij}} x_p \,, \quad (45)$$

where $P_{ij} = \{p : (i,j) \in A_p\}$ is the set of all paths traversing arc (i,j). Denoting, as before, by $a_{ij}(f_{ij})$ the cost of assuring the flow f_{ij} in the arc (i,j), we may formulate the optimization problem as:

$$\min_x \left[z(x) = \sum_{(i,j)\in A} a_{ij}(f_{ij}) = \sum_{(i,j)\in A} a_{ij}\left(\sum_{p \in P_{ij}} x_p \right) \right], \quad (46)$$

$$\sum_{p \in P_w} x_p = r_w, \quad \forall w \in W \,, \quad (47)$$

$$x_p \geq 0, \quad \forall p \in P_w, \quad \forall w \in W \,. \quad (48)$$

It turns out [4, 10], that the optimal distribution of path flows may be obtained by distributed partially asynchronous iterations of path flows x_p, grouped with respect to the realized connections w. The so-called "minimum first-derivative length" (MFDL) principle is applied. It says, that we should allocate more traffic to this path from the set P_w, for which the partial derivative of the cost function $\sum_{(i,j)\in A_p} \frac{\partial a_{ij}}{\partial x_p}$ is minimal. Applying the Taylor expansion series, it can be easily proved, that it guarantees for small amounts of shifted flow the decrease of the total cost. The assessment of MFDL path may be performed locally for every connection, that is in a distributed way, and asynchronously.
Since the information on flows in different arcs $(i,j) \in A_p$ for $p \in P_w$ comes from different times (e.g., the data concerning closer nodes is more recent) the wth processor, which calculates path flows of the wth connection, actually uses an estimate $\tilde{f}_{ij}^w(t)$ of these flows in some time window before the time of calculations t:

$$\tilde{f}_{ij}^w(t) = \sum_{\tau=t-B}^{t} \eta_{ij}^w(t,\tau) f_{ij}(\tau) \,, \quad (49)$$

where $f_{ij}(\tau)$ is the actual flow at time τ in the arc (i,j), B is the length of the time window, and $\eta_{ij}^w(t,\tau)$ are (usually unknown) nonnegative coefficients such that:

$$\sum_{\tau=t-B}^{t} \eta_{ij}^w(t,\tau) = 1 \,. \quad (50)$$

Let us denote the estimate of the derivative of the cost of the flow along the path $p \in P_w$ calculated at time t by $\lambda_p(t)$, that is:

$$\lambda_p(t) = \sum_{(i,j)\in A_p} a'_{ij}(\tilde{f}_{ij}^w(t)) \quad (51)$$

and the index of the MFDL path by p_m, that is:

$$\lambda_{p_m}(t) = \min_{p \in P_w} \lambda_p(t) \,. \quad (52)$$

In the general model it is assumed, that flows are not changed immediately and two phases are distinguished: the calculation of desired flows $\bar{x}_p$ and their realization x_p. According to this model, the new (actual) routing $x_p(t+1)$, $p \in P_w$ is determined as a convex combination of the desired routing $\bar{x}_p(t)$ and the current one $x_p(t)$:

$$x_p(t+1) = \beta_p(t)\bar{x}_p(t) + (1-\beta_p(t))x_p(t), \;\; p \in P_w \,, \quad (53)$$

where $0 < \beta < \beta_p(t) \leq 1$ are generally unknown coefficients reflecting a smooth (with geometric rate) movement from the current to the desired routing. Of course whichever they are, the transmission rate constraints (47) have to be satisfied.

The desired flows $\bar{x}_p$ for all paths in the connection w are calculated differently for the MFDL path and for the remaining ones. For paths $p \neq p_m$ the following formula is used:

$$\bar{x}_p(t) = \max\left\{0, x_p(t) - \frac{\gamma}{H_p(t)}\left(\lambda_p(t) - \lambda_{p_m}(t)\right)\right\}, \quad (54)$$

where $\gamma > 0$ is a stepsize and $H_p(t)$ is an estimate of the second derivative length of path p

$$H_p(t) = \sum_{(i,j)\in A_p} a''_{ij}(\tilde{f}^w_{ij}).$$

Afterwards, for the MFDL path the desired flow is calculated from the expression:

$$\bar{x}_{p_m}(t) = r_w - \sum_{p\in P_w, p\neq p_m} \bar{x}_p(t). \quad (55)$$

It may be proved, that there exists some $\gamma_0(B)$ such that for $0 < \gamma < \gamma_0(B)$ the described algorithm implemented asynchronously converges, delivering the minimum total cost of transmission $z(x)$. Luo and Tseng [8] showed, that when the cost function a_{ij} (e.g., the expected delay) on each link is a strictly convex function on the link flow, the sequence generated by this algorithm converges in the space of path flows at a linear rate.

It is possible to apply instead of (54) and (55) another scaled gradient algorithm:

$$\bar{x}_w(t+1) = \left[\bar{x}_w(t) - \gamma M_w^{-1}\lambda_w(t)\right]^+_{M_w(t)}, \quad (56)$$

where $\bar{x}_w$, λ_w are vectors formed of components $\bar{x}_p$, λ_p for $p \in P_w$, $M_w(t)$ is a symmetric positive definite matrix (usually it is an estimate of the Hessian matrix $\frac{\partial^2 z}{\partial x_w^2}$, and the algorithm (56) is an approximation of the projected Newton method), $[.]^+_{M_w(t)}$ denotes the projection on the simplex

$$\left\{x_w \middle| \sum_{p\in P_w} x_p = r_w \text{ and } x_p \geq 0, \forall p \in P_w\right\} \quad (57)$$

with respect to the norm $||x_w||_{M_w(t)} = \left(x'_w M_w(t) x_w\right)^{\frac{1}{2}}$. However, since this algorithm takes into account the current value of the desired flows $\bar{x}_w(t)$ instead of the current value of the actual flows $x_w(t)$ one may expect that it will be slower in the adaptation to sudden changes in the problem data r_w. Surprisingly, the replacement in Eq. (56) $\bar{x}_w(t)$ with $x_w(t)$ destroys the descent property and the convergence of the algorithm [4].

Recently Elwalid *et al.* [5] noticed, that the above scheme may be successfully adapted to Internet traffic engineering in multiprotocol label switching (MPLS) networks. They introduced two changes:

- They do not distinguish between the actual $x_p(t)$ and the desired $\bar{x}_p(t)$ source rates, that is a new rate vector is calculated from the formula:

$$x_w(t+1) = \left[x_w(t) - \gamma\lambda_w(t)\right]^+, \quad (58)$$

  where $[.]^+$ denotes the projection on the feasible space Eq. (57) with respect to the Euclidean norm. The justification is, that if one is only dealing with IP datagrams it is reasonable to assume that each ingress node can shift its traffic among the label switched paths available to it immediately after each update.

- They relax the assumption that at time t each source has available the current first derivative lengths Eq. (51) and uses it in place of the gradient in the update algorithm. Instead, they assume, that at time t, the source may only have outdated first derivative lengths. Moreover, the source uses a weighted average over several past lengths in the update algorithm. That is, the price used in algorithm (58) is calculated in the following way:

$$\lambda_p(t) = \sum_{\tau=t-B}^{t} \sum_{(i,j)\in A_p} \rho^w_{ij}(t,\tau)\, a'_{ij}(\tilde{f}^w_{ij}(\tau)), \quad (59)$$

  where $\tilde{f}_{ij}(\tau)$ is an estimate of flow in the arc (i,j) calculated at time τ Eq. (49), B is the length of the time window, and $\rho^w_{ij}(t,\tau)$ are (usually unknown) nonnegative coefficients such that:

$$\sum_{\tau=t-B}^{t} \rho^w_{ij}(t,\tau) = 1. \quad (60)$$

  This is because, in the distributed and decentralized implementation of the algorithm, the source can only estimate the first derivative lengths through noisy measurement.

Despite these differences, stability of this algorithm (called MATE – from multipath adaptive traffic engineering) in [5] has been established using the same techniques as in [4, 10].

4. Conclusions

All presented distributed, asynchronous optimization methods for data networks management may be interesting to network operators. For mass client market traffic balance routing (see Section 2) may be sufficient. For more demanding users: state services, governmental institutions, big companies, banks, etc., the model with guaranteed connection rates (see Section 3) should be applied. While in the first case prices, i.e., Lagrange multipliers, are only some internal indicators guiding the network towards the optimum and the balance of resources and demands, without the monetary consequences, in the second case they may be more useful. Namely, it is possible to use them directly to calculate online the cost of high-quality connections or to draw up a new price list for future SLAs.

References

[1] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*. New York [etc.]: Wiley, 1993.

[2] P. Beraldi, F. Guerriero, and R. Musmanno, "Parallel algorithms for solving the convex minimum cost flow problem", *Comput. Opt. Appl.*, vol. 18, pp. 175–190, 2001.

[3] D. P. Bertsekas, *Network Optimization: Continuous and Discrete Models*. Belmont: Athena Scientific, 1998.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont: Athena Scientific, 1997.

[5] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: multipath adaptive traffic engineering", *Comput. Netw.*, vol. 40, issue 6, pp. 695–709, 2002.

[6] W. Findeisen, F. N. Bailey, M. Brdyś, K. Malinowski, and A. Woźniak, *Control and Coordination in Hierarchical Systems*. Chichester [etc.]: Wiley, 1980.

[7] A. Karbowski, "Distributed asynchronous algorithms in the Internet – new routing and traffic control methods", *J. Telecomm. Inform. Technol.*, no. 3, pp. 29–36, 2005.

[8] Z.-Q. Luo and P. Tseng, "On the rate of convergence of a distributed asynchronous routing algorithm", *IEEE Trans. Automat. Contr.*, vol. 39, issue 5, pp. 1123–1129, 1994.

[9] P. Tseng, D. P. Bertsekas, and J. N. Tsitsiklis, "Partially asynchronous algorithms for network flow and other problems", *SIAM J. Contr. Opt.*, vol. 28, pp. 678–710, 1990.

[10] J. N. Tsitsiklis and D. P. Bertsekas, "Distributed asynchronous optimal routing in data networks", *IEEE Trans. Automat. Contr.*, vol. AC-31, no. 4, pp. 325–332, 1986.

Andrzej Karbowski received M.Sc. degree in electronic engineering (specialization automatic control) from Warsaw University of Technology (Faculty of Electronics) in 1983. He received Ph.D. in 1990 in automatic control and robotics. He works as adjunct both at Research and Academic Computer Network (NASK) and at the Faculty of Electronics and Information Technology (at the Institute of Control and Computation Engineering) of Warsaw University of Technology. His research interests concentrates on data networks management, optimal control in risk conditions, decomposition and parallel implementation of numerical algorithms.

e-mail: A.Karbowski@ia.pw.edu.pl
Research and Academic Computer Network (NASK)
Wąwozowa st 18
02-796 Warsaw, Poland

Institute of Control and Computation Engineering
Warsaw University of Technology
Nowowiejska st 15/19
00-665 Warsaw, Poland