Asynchronous Network Coded Multicast

Danail Traskov, Johannes Lenz
Institute for Communications Engineering
Technical University Munich, Germany
danail.traskov@tum.de, johannes.lenz@mytum.de

Niranjan Ratnakar
Qualcomm
Bedminster, NJ

Muriel Médard
Research Laboratory of Electronics
MIT, USA
medard@mit.edu

***Abstract*—We consider the problem of setting up a multicast connection of minimum cost using network coding. It is well-known that this can be posed in the form of a convex program. Our contribution is an asynchronous algorithm for solving the optimization problem, in analogy to the well-known distributed asynchronous Bellman-Ford algorithm for routing. Furthermore, we provide extensive simulation results showing fast convergence despite the lack of any central clock in the network and robustness with respect to link- or node failures.**

I. Introduction

For multicasting in networks random linear network coding is a capacity achieving scheme [1]. For a given multicast session it is often desirable to find a solution that can satisfy the demands while minimizing ressource consumption in the network, such as bandwith or energy. This is equivalent to identifying a subnetwork that, fully utilized, can accommodate the demands. The authors in [2] have suggested posing the subgraph selection problem as a linear or convex program, depending on which resource usage is to be minimized. Furthermore, they propose distributed algorithms for solving the resulting optimization problems, subgradient optimization for the linear case and a primal-dual algorithm for the convex case. These algorithms assign to each network node a processor and find the optimal solution by solely exchanging update messages between processors that are neighbors in the network.

Such a distributed operation is highly desirable in practice, as otherwise the whole network topology has to be collected at a special "fusion center" which carries out the computation and then the calculated link- and injection rates have to be communicated across the network, leading to significant overhead. Still, one central assumption in these algorithms is that in every time slot the updates at all nodes are carried out simultaneously, as if triggered by a central clock signal [2], [3]. The contribution of our work is to relax this assumption and instead propose an *asynchronous* algorithm for solving the problem. As we show, our approach gets by with very small restrictions on the update schedule, even a completely random sequence of node updates will converge, as long as each node is chosen with non-zero probability.

Distributed asynchronous algorithms have been proposed and used for routing, the most prominent example being the distributed Bellman-Ford algorithm, which has both a synchronous and an asynchronous variation [4, Section 5.2.4]. For network coded traffic and the particular optimization problem associated with it [2] this is, to the best of our knowledge, the first asynchronous approach. The asynchronous algorithm proposed in [5] addresses the problem of setting up a network code and is therefore orthogonal to our approach, as we are concerned with providing and allocating a minimal set of network ressources, before network coding is carried out on top of these ressources.

The motivations for seeking asynchronous solutions are two-fold. Firstly, in large networks the assumption of having a clock that is available at each node is unrealistic or requires a significant amount of communication overhead. The fundamental limits of clock synchronization across a network are discussed in [6] and the references therein. Secondly, network transmissions often have a non-negligible loss- or error rate. When the algorithm requires synchronous round of updates such losses of messages can seriously impair convergence. An asynchronous algorithm as the one we suggest, on the other hand, can easily deal with lost update messages due to the minimal requirements it poses on the update schedule.

The main idea of our work is to apply a block-coordinate ascent algorithm to the convex program that describes the multicast connection. This algorithm, as we show by means of simulations, has fast convergence compared to the primal-dual algorithm in [2] and is very robust with respect to randomly occurring node updates. Even in the presence of link failures, the algorithm continues updating and eventually converges without the need of a restart.

The rest of the paper is organized as follows. In Section II, we review the network model and the problem formulation of the convex program. In Section III, we derive the asynchronous algorithm and prove its convergence, while the experimental results are presented in Section IV. Finally, in Section V we conclude the paper.

II. Network Model and Optimization Problem

The network is represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{A}$ is the set of directed arcs. A multicast session $(s, \mathcal{T}, R)$ is a triple consisting of a source node s a destination set $\mathcal{T} \subset \mathcal{V}$ and a rate R. Let T denote the cardinality of $\mathcal{T}$. All destinations are interested in the same information from the origin at rate R (as opposed to a multi-level multicast problem, where some sources need only a subset of the information [7]). Another extension which is rather straightforward is multi-source multicasting, where we have several sources, but every destination is interested in the entire information originating at *every* source. This extension has been pursued in [8]. For each destination $t \in \mathcal{T}$ we

index the associated flow (and later price) variables with the superscript t.

The following convex program [2] establishes a single multicast connection in a network at rate R:

$$\text{minimize} \sum_{(i,j)\in\mathcal{A}} f_{ij}(z_{ij})$$

subject to

$$\sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{(t)} - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{(t)} = \sigma_i^{(t)}, \quad \forall i \in \mathcal{V}, t \in \mathcal{T}, \quad (1)$$

$$0 \le x_{ij}^{(t)} \le z_{ij}, \qquad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T}, \quad (2)$$

$$(z_{ij})_{j:(i,j)\in\mathcal{A}} \in \mathcal{C}_i, \qquad \forall i \in \mathcal{V}, \quad (3)$$

where

$$\sigma_i^{(t)} = \begin{cases} R & i = s, \\ -R & i = t, \\ 0 & \text{else}, \end{cases} \quad (4)$$

and $\mathcal{C}_i$ is a convex subset of the positive orthant containing the origin. The equations in (1) establish flows at rate R from the source to all multicast sinks. Constraint (2) means that the actual link usage is the maximum value of the flows traversing it. This is where network coding enters the picture; with routing the actual link usage would be simply the sum of flows going across. Finally, (3) models the physical layer capacity region $\mathcal{C}_i$ at node i, and can be given by a wireline network capacity constraint or an arbitrary broadcast channel [9].

III. Decentralized Asynchronous Algorithm

In this section we apply a block coordinate ascent algorithm to the convex program (1)-(3) resulting in an, as we show, decentralized and asynchronous operation. To be concrete, assume that the capacity regions $\mathcal{C}_i$ are given by $z_{ij} \in [0, c_{ij}]$, where the link capacities c_{ij} are fixed constants, as for example in wireline networks. More complicated models have been studied in [8].

Since the function $z_{ij} = \max_{t\in\mathcal{T}} x_{ij}^{(t)}$ is not differentiable, it poses a challenge for gradient-type optimization algorithms. One approach is to replace this relation with a soft-maximum that is differentiable and that approximates the maximum function. Two common approximations are the *log-sum-exp* function [10] and the l_p-norm [11], given by

$$z'_{ij} = L \log \left(\sum_{t\in\mathcal{T}} \exp\left(x_{ij}^{(t)} / L \right) \right), \quad (5)$$

and

$$z''_{ij} = \left(\sum_{t\in\mathcal{T}} \left(x_{ij}^{(t)} \right)^p \right)^{1/p}, \quad (6)$$

respectively. Both functions converge to the maximum function, for $L \to 0$ and for $p \to \infty$, respectively. Although they are convex and differentiable, they are not *strictly* convex. This can be seen by setting $x_{ij}^{(t)} = x_{ij}, \quad \forall t \in \mathcal{T}$. For the *log-sum-exp* function, this leads to $z'_{ij} = L \log T + x_{ij}$, which is linear in x_{ij} and therefore not strictly convex. Replacing z_{ij} with z'_{ij} we can define a modified cost function F_{ij} according to

$$F_{ij}(\underline{x}_{ij}) = f_{ij}\left(L \log \left(\sum_{t\in\mathcal{T}} \exp\left(x_{ij}^{(t)} / L \right) \right) \right), \quad (7)$$

where $\underline{x}_{ij} = \left(x_{ij}^{(1)}, ..., x_{ij}^{(T)} \right)$. $F_{ij}(\underline{x}_{ij})$ is (strictly) convex if $f_{ij}(\cdot)$ is (strictly) convex and monotonically increasing. In this work, we shall assume all cost functions $f_{ij}(\cdot)$ to be strictly convex and monotonically increasing, thus giving rise to a strictly convex objective function. With this transformation the problem is reformulated as a standard convex multi-commodity flow problem, where the flows are coupled only via the cost function and the constraints are separable [12, Section 8.3]. The primal optimization problem takes on the form

$$\text{minimize} \sum_{(i,j)\in\mathcal{A}} F_{ij}(\underline{x}_{ij})$$

subject to

$$\sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{(t)} - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{(t)} = \sigma_i^{(t)}, \qquad \forall i \in \mathcal{V}, t \in \mathcal{T}, \quad (8)$$

$$0 \le x_{ij}^{(t)} \le c_{ij}, \qquad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T}. \quad (9)$$

Introducing a Lagrange multiplier $p_i^{(t)}$ for every constraint in (8), we form the Lagrangian

$$L(x,p) = \quad (10)$$

$$= \sum_{(i,j)\in\mathcal{A}} F_{ij}(\underline{x}_{ij}) \quad (11)$$

$$+ \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{V}} p_i^{(t)} \left(\sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{(t)} - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{(t)} - \sigma_i^{(t)} \right) \quad (12)$$

$$= \sum_{(i,j)\in\mathcal{A}} \left(F_{ij}(\underline{x}_{ij}) + \sum_{t\in\mathcal{T}} \left(x_{ij}^{(t)} \left(p_i^{(t)} - p_j^{(t)} \right) \right) \right) \quad (13)$$

$$- \sum_{i\in\mathcal{A}} \sum_{t\in\mathcal{T}} p_i^{(t)} \sigma_i^{(t)}. \quad (14)$$

The dual function value $q(p)$ at a price vector p is

$$q(p) = \sum_{(i,j)\in\mathcal{A}} g_{ij}(\underline{p}_i - \underline{p}_j) - \sum_{i\in\mathcal{A}} \sum_{t\in\mathcal{T}} p_i^{(t)} \sigma_i^{(t)},$$

where $g_{ij}(\cdot)$ is defined as

$$g_{ij}(\underline{p}_i - \underline{p}_j) =$$

$$= \inf_{0 \le \underline{x}_{ij} \le c_{ij}} \left\{ F_{ij}(\underline{x}_{ij}) + \sum_{t\in\mathcal{T}} \left(x_{ij}^{(t)} \left(p_i^{(t)} - p_j^{(t)} \right) \right) \right\}, \quad (15)$$

and $\underline{p}_i = \left(p_i^{(1)}, ..., p_i^{(T)}\right)$. The solution of the dual unconstrained optimization problem

$$\max_{p} \; q(p)$$

is equivalent to the solution of the primal problem as under our assumptions there is no duality gap. We suggest solving the dual by a block coordinate ascent method. To that end, consider the $|\mathcal{V}|$ variable blocks $\underline{p}_i$. At the k-th iteration a block $\underline{p}_i$ is selected and the maximization as carried out with respect to the variables $\underline{p}_i$. We defer the discussion of how to select blocks in order to achieve convergence to the end of the section. The update takes on the form

$$\underline{p}_i[k+1] = \underset{\underline{p}_i[k]}{\operatorname{argmax}} \sum_{(i,j)\in\mathcal{A}} g_{ij}(\underline{p}_i[k] - \underline{p}_j[k]) - \sum_{j\in\mathcal{A}} \sum_{t\in\mathcal{T}} p_j^{(t)}[k]\sigma_j^{(t)}. \tag{16}$$

How this maximization can be carried out is an intricate issue, since, in general, $g_{ij}(\cdot)$ is not available in closed form but only from the expression (15). There is, however, some structure that we can exploit. The key is the following Lemma, which is a special case of [13, Prop. 6.1.1]

Lemma 1: Let $x(p)$ be the unique minimizer of the Lagrangian at a price vector p, i.e.

$$x(p) = \operatorname*{argmin}_{x} L(x, p). \tag{17}$$

Then, the dual function $q(p)$ is everywhere continously differentiable and its derivative in the p_i^t-coordinate is given by the constraint function evaluated at $x(p)$, in our case this is

$$\frac{\partial q}{\partial p_i^t} = \sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{(t)}(p) - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{(t)}(p) - \sigma_i^{(t)}. \tag{18}$$

Remark 1: In other words, the derivative is given by the flow divergence out of node i for each session t.

Remark 2: Linear constraints and a *strictly* convex objective function (as we have assumed throughout) imply the uniqueness of the Lagrangean minimizer and therefore the validity of the Lemma. Mere convexity is not sufficient, in general.

An update at node i takes on the following form. With every edge adjacent to i we associate a processor that solves problem (15) and computes the minimizer. For an edge (i,j) this becomes

$$\underline{x}_{ij}(\underline{p}_i - \underline{p}_j) = = \underset{0\le x_{ij}\le c_{ij}}{\operatorname{argmin}} \left\{ F_{ij}(\underline{x}_{ij}) + \sum_{t\in\mathcal{T}} \left(x_{ij}^{(t)} \left(p_i^{(t)} - p_j^{(t)} \right) \right) \right\}, \tag{19}$$

and requires the price vectors of the neighboring nodes only. As this optimization is convex, it can be solved with standard algorithms like SQP [14] or a Projected Gradient method [13]. Owing to the simple constraints on $x_{ij}^{(t)}$ the orthogonal projection can be implemented easily. Node i gathers the $x_{ij}^{(t)}(p)$ from adjacent edges to compute the gradient with respect to $p_i^{(1)}, \ldots, p_i^{(T)}$ in the following way

$$\nabla_i = \begin{pmatrix} \sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{(1)}(p) - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{(1)}(p) - \sigma_i^{(1)} \\ \vdots \\ \sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{(T)}(p) - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{(T)}(p) - \sigma_i^{(T)} \end{pmatrix}, \tag{20}$$

and updates the prices according to

$$\underline{p}_i := \underline{p}_i + \alpha\nabla_i. \tag{21}$$

To prove convergence of the described algorithm we need to specify the sequence in which the variable blocks are updated. Consider the following definition [15]:

Definition 1: We speak of a *partially asynchronous order* if there exists positive constant K for which every coordinate is chosen at least once for relaxation between iterations r and $r+K,\ r = 0, 1, ...$ Furthermore, at any iteration the varibales used to compute the update are at most K steps old.

Remark 3: For the choice of $K = |\mathcal{V}|$ this leads to a cyclical update rule, while for K large it comes close to a random choice of the current block of variables.

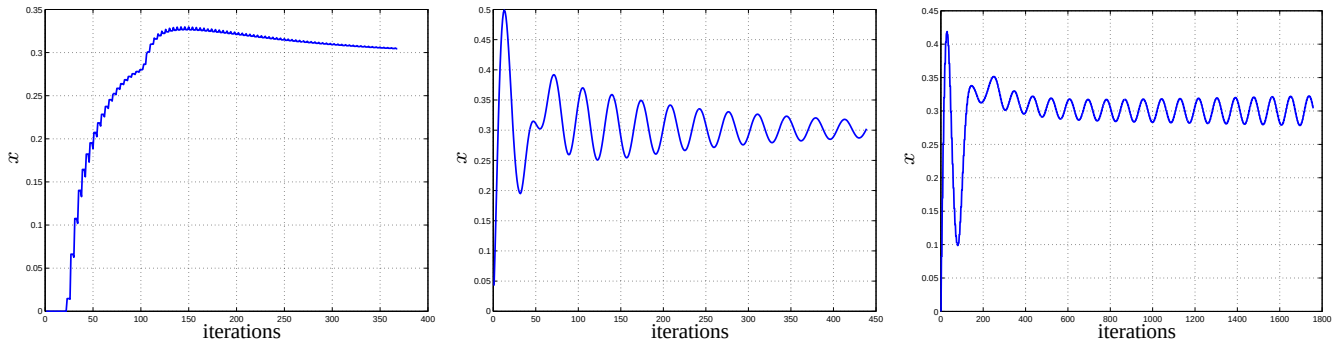
Under such an order, [15, Proposition 5.2] has proven the convergence of block coordinate ascent.

Proposition 1: If the function $q(p)$ is continuously differentiable, the gradient satisfies a Lipschitz-condition, and a partially asynchronous update order is adopted block coordinate ascent converges for a stepsize sufficiently small.

Remark 4: The algorithm converges (under some more technical conditions) even in the case that an arbitrary descent direction is used in place of the gradient [15, Section 7.5].

The stepsize α can be determined by standard line search algorithms like Backtracking [10] or the Golden Section Method [13] . Note, that (19) is a very low dimensional problem - its dimension is the number of multicast sinks T - and can be solved readily a number of times. The complexity of a node update scales proportionally to the complexity of (19), the number of adjacent edges of i and the number of steepest ascent iterations carried out.

From the algorithm description, we see that all steps, i.e. the computation of the resulting flow for a fixed price vector (19), the computation of the gradient (20) and the price update (21) require solely information that can be gathered in a *one-hop* neighborhood of the updating node. This gives rise to the decentralized operation of the algorithm. If combined with an essentially cyclic update order, assuming a large constant K, we also conclude asynchronous convergence.



(a) (b) (c)

Fig. 2. Progress of the asynchronous block descend algorithm (left), synchronous primal-dual algorithm (center) and asynchronous primal-dual algorithm (right). We measure convergence with respect to the flow x, of which the optimal value is 0.3.

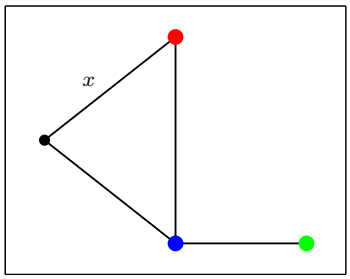


Fig. 1. A network where the top node multicasts to the two bottom nodes.

IV. Performance Evaluation

To assess empirically the performance of our approach we conduct a series of experiments. The link cost function is taken to be throughout $a_{ij}\exp(z'_{ij})$, where a_{ij} is a positive coefficient. First, we illustrate a case where, owing to asynchronous updates, the primal-dual algorithm of [2] fails to converge, but our algorithm, as theory suggests, converges. Consider Fig. 1, and the convergence curves in Fig. 2. The primal-dual algorithm converges correctly if applied synchronously, but fails to converge if updated asynchronously (the updates are carried out in cyclical order, or $K = 4$). Another interesting observation is that coordinate ascent and synchronous primal-dual converge after a similar number of iterations; in the primal-dual, however, during an iteration *every* node in the network performs an update, as compared to just one node in the network for the coordinate ascent. This has two implications: Firstly, coordinate ascent needs less communication overhead, reducing control traffic in the network. Secondly, if for larger networks some of the updates could be carried out in parallel, this would lead to a significant speed-up.

Figures 3(a) - 3(e) illustrate the convergence of the block coordinate ascent algorithm in a larger randomly generated unit disc graph. We see the topology in Fig. 3(a), and the flows after convergence in Fig. 3(b). In Fig. 3(c), we plot the value of the dual function $q(p)$ (normalized to 1) for the coordinate ascent and a random selection of updating nodes. In comparison, we plot the value of the Lagrangian for the primal-dual algorithm of [2], when updated in synchronous rounds. In Figs. 3(d) and 3(e), we show the primal convergence for two selected flows. Note, that the dual optimum is approximated much faster than the primal, a result consistent with [2], [3].

Finally, we investigate the behaviour under dynamic conditions, when a link in the network suddenly fails. Figure 4 shows the reaction of the algorithm to the failure of one edge. Since the price variables p are unconstrained, every value can be used as a starting point of the algorithm. Consequently, the coordinate ascent algorithm converges in the event of network changes without any modifications, particularly without the need of a restart.

V. Conclusion

We have proposed a fully decentralized and asynchronous algorithm for the minimum-cost multicast problem. Simulations show fast convergence as compared to previously known approaches and robustness as no assumption on network synchronization is made. This holds even in the presence of topology changes such as link failures. A worthwhile direction of further work would be to investigate the potential for carrying out some of the update steps in parallel and therefore speeding up the algorithm even further.

References

[1] T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D. Karger, "A random linear network coding approach to multicast," *IEEE Trans. on Information Theory*, vol. 52 (10), pp. 4413–4430, 2006.

[2] D. Lun, N. Ratnakar, M.Médard, R. Koetter, D. Karger, T. Ho, and E. Ahmed, "Minimum-cost multicast over coded packet networks," *IEEE Trans. on Information Theory*, June 2006.

[3] F. Zhao, M. Médard, . D. S. Lun, and A. Ozdaglar, "Convergence rates of min-cost subgraph algorithms for multicast in coded networks," *Proceedings of Allerton Annual Conference on Communication, Control and Computing*, 2007.

[4] D. Bertsekas and R. Gallager, *Data networks (2nd ed.)*. Upper Saddle River, NJ: Prentice-Hall, Inc., 1992.

[5] T. Ho, B. Leong, R. Koetter, and M. Médard, "Distributed asynchronous algorithms for multicast network coding," *1st Workshop on Network Coding, WiOpt*, 2005.

[6] N. Freris and P. Kumar, "Fundamental limits on synchronization of affine clocks in networks," in *Decision and Control, 2007 46th IEEE Conference on*, Dec. 2007, pp. 921–926.

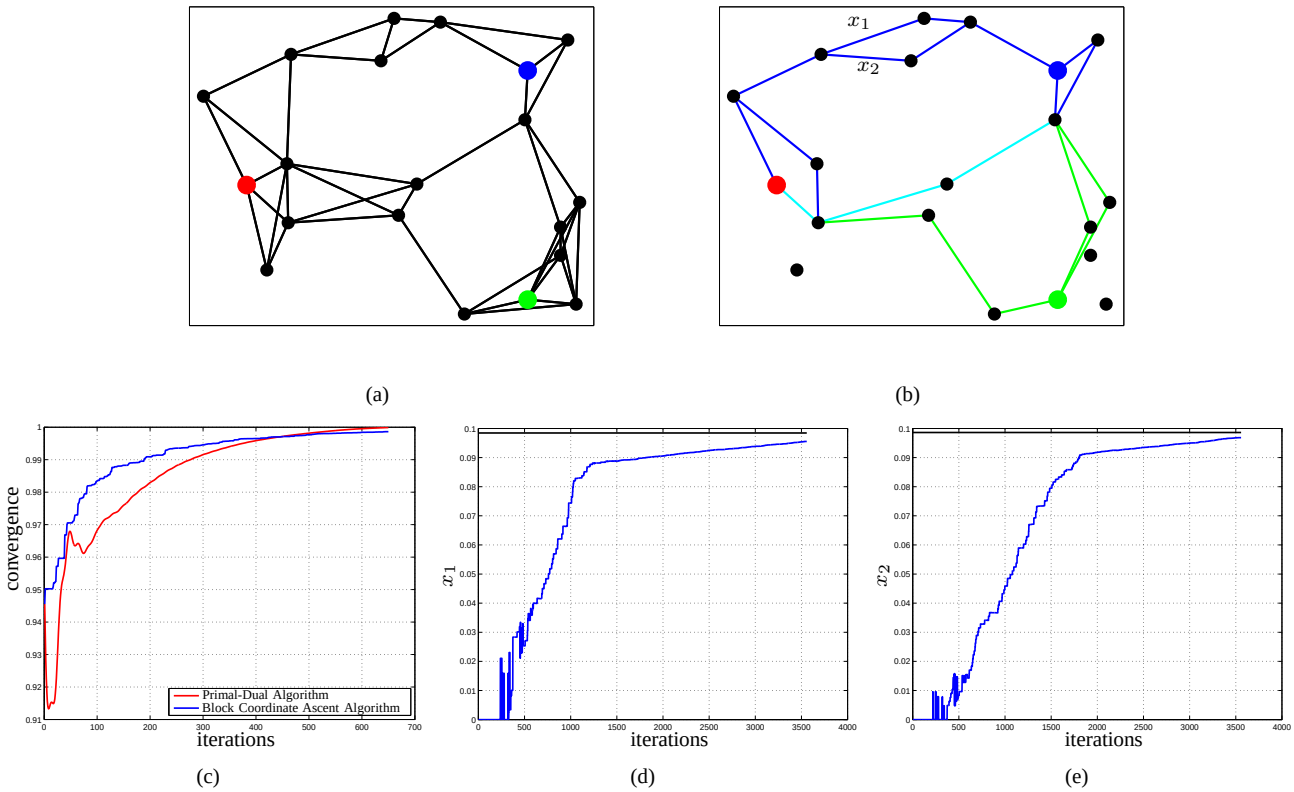


Fig. 3. Top: Network topology (left) and corresponding flows after convergence (right). The source is red, sinks are blue and green. The network coded flow to both sinks is cyan. Bottom: (left) Convergence of coordinate ascent (asynchronously) and primal-dual (synchronous rounds). In the (center) and (right) the convergence of the link flows x_1 and x_2. The black line denotes the optimal flow value

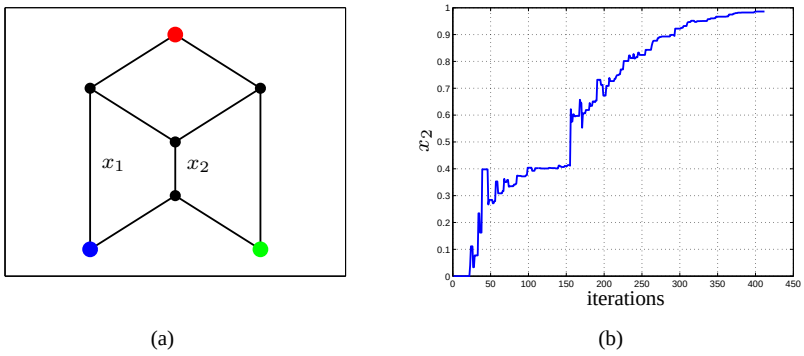


Fig. 4. In the network topology (left), the edge carrying flow x_1 fails after 150 iterations. (Right) The flow on x_2 first converges to the optimal value of 0.4. After the link failure it converges to the new optimal value of 1.

[7] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 782–795, 2003.

[8] J. Lenz, "Asynchronous network coded multicast," *MS thesis*, 2009.

[9] T. Cover and J. Thomas, *Elements of Information Theory*. Wiley, 1991.

[10] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge University Press, 2004.

[11] S. Deb and R. Srikant, "Congestion control for fair resource allocation in networks with multicast ows," *IEEE/ACM Trans. Networking*, vol. 12, no. 2, pp. 274–285, April 2004.

[12] D. Bertsekas, *Network Optimization - continuous and discrete models*. Belmont, MA: Athena Scientific, 1998.

[13] ——, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1995.

[14] P. Boggs and J. Tolle, "Sequential quadratic programming," *Acta Numerica*, pp. 199–242, 1995.

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