INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Distributed Gradient Optimization for Epidemic Routing: a preliminary Evaluation

Giovanni Neglia — Giuseppe Reina — Sara Alouf

N° 7016

August 2009

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--7016--FR+ENG

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
INRIA
centre de recherche SOPHIA ANTIPOLIS - MÉDITERRANÉE

Distributed Gradient Optimization for Epidemic Routing: a preliminary Evaluation

Giovanni Neglia * , Giuseppe Reina * , Sara Alouf *

Thème COM — Systèmes communicants
Équipes-Projets Maestro

Rapport de recherche n° 7016 — August 2009 — 14 pages

Résumé : Dans ce rapport, nous nous intéressons aux réseaux tolérant les délais et, plus particulièrement, à la transmission de messages dans ce type de réseaux. L'objectif est de concevoir des mécanismes adaptatifs, qui relaieraient les messages de proche en proche jusqu'à leurs destinations. Notre approche repose sur des résultats récents en optimisation multi-agent, qui mettent en œuvre des méthodes de sous-gradient distribuées. Nous cherchons comment de telles méthodes pourraient être appliquées dans le contexte des réseaux tolérant les délais et menons une étude préliminaire sur leurs performances. Les résultats obtenus sont encourageants, en particulier ceux concernant la vitesse de convergence vers la solution optimale qui minimise une fonction de coût prédéfinie.

Mots-clés : Réseaux tolérant les délais, routage épidémique, optimisation en ligne, optimisation distribuée par descente de gradient

* INRIA, 2004 Route des Lucioles, Sophia-Antipolis, France, email: name.surname@sophia.inria.fr

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Distributed Gradient Optimization for Epidemic Routing: a preliminary Evaluation

Abstract: In this research report we address the problem of designing adaptive epidemic-style forwarding mechanisms for message delivery in Delay Tolerant Networks. Our approach is based on a new analytical framework for multi-agent optimization through distributed subgradient methods. We investigate how this framework can be adapted to the considered networking problem and we perform a preliminary evaluation, which shows promising results in terms of convergence speed.

Key-words: Delay tolerant networks, epidemic routing, online optimization, distributed gradient optimization

Table des matières

1 Introduction

Delay Tolerant Networks (DTNs) are sparse and/or highly mobile wireless ad hoc networks where no continuous connectivity guarantee can be assumed [1, 2]. One central problem in DTNs is related to the routing of packets towards the intended destination. Protocols developed in the mobile ad hoc networks field, indeed, fail since a complete route to the destination may not exist most of the time. One common technique for overcoming such problem is to disseminate multiple copies of the message in the network, enhancing the probability that at least one of them will reach, within a suitable time-frame, the destination node [3]. This is referred to as epidemic-style forwarding [4]. Alike the spread of infectious diseases, each time a message-carrying node encounters a new node not having a copy thereof, it may *infect* this new node by passing on a message copy ; newly infected nodes, in turn, may behave similarly. The destination receives the message when it meets an infected node.

An unconstrained epidemic forwarding scheme (in which an infected node spreads messages to all nodes it encounters) is able to achieve minimum delivery delay at the expense of an increased use of resources such as buffer space, bandwidth, and transmission power. Variations of epidemic forwarding have been recently proposed in order to exploit the trade-off between delivery delay and resource consumption. This family includes, among others, K-hop schemes [5], probabilistic forwarding [6], and spray-and-wait [7]. These schemes differ in their "infection process," i.e., the spreading of a message in network.

Depending on the specific application scenario, different performance metrics could be envisaged, such as the probability to successfully deliver a message to destination, the delivery time, the total energy consumption in the system or a combination of the previous ones. Optimal policies have been identified for specific metrics in restricted set of policies. For example [8] and [9] focus on 2-hop schemes (only the source can copy the message and infected nodes act as relays to the destination) and derive optimal configurations for the two following cases : 1) the source can select the maximum number of copies that minimizes a weighted sum of the delivery time and the energy consumption [8] ; 2) the source copies a message with a (time-dependent) probability to maximize the delivery probability under energy consumption constraints [9]. In both cases, parameters configuration depends on the specific network scenario, e.g., on the number of nodes in the system and on their mobility patterns. In many cases, these characteristics may not be known at system design and deployment time

and may drastically change across time and space. Consider for instance a personal digital assistant (PDA) carried by a user in its daily activities. During the day, the PDA may travel at different speeds (e.g. from zero up to car speed), moving from highly crowded areas (supermarkets, classrooms, etc.) to sparse ones, with very different trajectories (straight along a highway or following a random walk from shop to shop) and different levels of power availability. Interestingly, [9] proposes also an algorithm based on stochastic approximation for online learning. Nevertheless the problem of deriving optimal forwarding policies for general optimization goals and in large sets of possible behaviours is mainly open.

For this reason, our interest is to develop online adaptive policies able to optimize generic performance metrics. A previous attempt in such direction has been carried on by some of the authors who have applied concepts and tools from the Genetic Algorithms (GAs) field. Each node employs a (potentially different) forwarding policy, which prescribes the operations to be undertaken when receiving a message destined to another node. Such a policy is described by an array of parameters called the genotype. Genotypes are associated with a fitness measure which, roughly speaking, indicates the ability of the current set of parameters to achieve good performance in the local environment. Fitness is evaluated using local information and feedback which is sent from the destination backwards. Some results have been presented in [10] and in [11]. A criticism to this approach is that GAs (and evolutionary algorithms in general) are suited for problems with ill-behaved functions having multiple minima for which gradient based methods would fail. On the contrary, it seems natural to expect that network performance metrics of possible interest exhibit a more regular behaviour. For this reason, in this work, we start exploring how to implement a distributed gradient algorithm. Our approach is based on the analytical framework recently proposed in [12] : the authors study a distributed computation model for optimizing a sum of convex objective functions corresponding to multiple agents. The method involves every agent minimizing its own objective function while exchanging information locally with other agents in the network over a time-varying topology. The contribution of this work is twofold : first, we show how this approach can be used to optimize routing in a DTN, and second, we perform a preliminary evaluation in terms of convergence speed and quality of the solution identified.

The report is organized as follows. Section 2 provides the required background on the distributed subgradient method proposed in [12]. In Section 3, we show how this approach can be adapted to work in a DTN scenario and stress the key issues to be solved. Section 4 presents a preliminary evaluation of a case study and finally Section 5 concludes the report and illustrates future research directions.

2 Background on Distributed Gradient Methods

In [12], the authors study a distributed computation model for optimizing a sum of convex objective functions corresponding to m agents. The method involves every agent minimizing a *local convex* cost function $f_i : \mathbb{R}^n \to \mathbb{R}$ while exchanging information locally with other agents in the network over a deterministically time-varying topology.

The agents want to cooperatively solve the following unconstrained optimization problem :

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}}\, F(\mathbf{x}); \quad F(\mathbf{x}) := \sum_{i=1}^{m} f_i(\mathbf{x}). \tag{1}$$

Let F^* be the minimum value of F and S^* be the optimal solution set, i.e., for all $\mathbf{x}$ in S^*, $F(\mathbf{x}) = F^*$.

To solve (1), the distributed gradient method works on a time slot basis. At any point in time, each agent has available an estimate of an optimal solution of the problem (1). During a time slot, an agent can communicate its estimate to a subset of the other agents. We denote by $\hat{\mathbf{x}}^i(k)$ the estimate maintained by agent i at the k-th slot.

At the end of a time slot, each agent updates its estimate according to the following relation :

$$\hat{\mathbf{x}}^i(k+1) = \sum_{j=1}^{m} a_j^i(k)\hat{\mathbf{x}}^j(k) - \zeta \mathbf{d}_i(k), \tag{2}$$

where $\mathbf{a}^i(k)$ is an m-length vector of non-negative weights such that $\sum_{j=1}^{m} a_j^i = 1$, $\zeta > 0$ is a tunable step size and $\mathbf{d}_i(k)$ is a subgradient of the function f_i at the point $\hat{\mathbf{x}}^i(k)$. [1] In this report, we consider $\mathbf{d}_i(k)$ to be the gradient of the function f_i, i.e., $\mathbf{d}_i(k) = \nabla f_i(\mathbf{x})|_{\mathbf{x}=\hat{\mathbf{x}}^i(k)}$. Hence, the last addend in (2) acts in the direction of minimizing the function f_i (not F). At the same time, the first sum corresponds to averaging the estimate of all the other nodes, similarly to what is done in consensus protocols. In order to present the main results of [12] in a simple form, we consider that agent i sets the weights in the following way : if the estimate of agent j is available at agent $i \neq j$ during time slot k then $a_j^i(k) = 1/m$, otherwise $a_j^i(k) = 0$; last, $a_i^i(k) = 1 - \sum_{j\neq i} a_j^i(k)$.

In [12], it is shown that the update rule (2) leads the estimate $\hat{\mathbf{x}}^i(k)$ to converge *near* the optimal solution set if the following assumptions are satisfied :

- **Connectivity :** The information of each agent influences the information of any other agent infinitely often (eventually through a sequence of intermediate agents). Formally, consider the graph (V, E_∞), where V is the set of agents and E_∞ is the set of links such that link (i, j) exists if and only if i and j communicate directly infinitely often. Then, the connectivity assumption corresponds to the graph (V, E_∞) being connected.
- **Bounded intercommunication interval :** The time between two consecutive communications between two agents that communicate infinitely often is upper bounded. Formally, there exists a positive integer B such that for every $(i, j) \in E_\infty$, i and j exchange information status at least once every B consecutive time slots.
- **Bounded gradients :** There exists L such that $||\nabla f_i|| \leq L$.

Under the above assumptions, the authors of [12] prove that

$$F(\overline{\mathbf{x}}^i(k)) - F^* \leq \frac{m\ \mathrm{dist}^2(\mathbf{y}(0), S^*)}{2\zeta k} + \zeta L^2 C(m), \tag{3}$$

where $\overline{\mathbf{x}}^i(k) := \frac{1}{k}\sum_{h=0}^{k-1} \hat{\mathbf{x}}^i(h)$ is the time average vector, $\mathbf{y}(0) := \frac{1}{m}\sum_{i=1}^{m} \hat{\mathbf{x}}^i(0)$ is the average initial estimate, $\mathrm{dist}(\mathbf{y}(0), S^*)$ denotes the 2-norm distance between the vector $\mathbf{y}(0)$ and the set S^*, and

1. By definition of a subgradient, the vector $\mathbf{d}_i(k)$ satisfies $\mathbf{d}_i(k)(\mathbf{x} - \hat{\mathbf{x}}^i(k)) \leq f_i(\mathbf{x}) - f_i(\hat{\mathbf{x}}^i(k))$ for any $\mathbf{x}$ in $\mathbb{R}^n$.

$$C(m) = \frac{1}{2} + 6m + \\ + \frac{6m^2(1+m^{(m-1)B})/(1-1/m^{(m-1)B})}{1-\left(1-1/m^{(m-1)B}\right)^{\frac{1}{(m-1)B}}}.$$

Inequality (3) states that, at any time slot k, the distance between the function F evaluated at $\overline{\mathbf{x}}^i(k)$ and its minimum is bounded by the sum of two terms. The first of these terms converges to zero as k goes to infinity, but the second one is constant and can be reduced only by decreasing the step size ζ. We observe that the bound in (3) is not very tight, in fact $C(m)$ is larger than $m^{(m-1)B}$ which is very large already for a system with tens of nodes. This observation also justifies our numerical analysis in Section 4.

3 Application to Epidemic Routing

In this section, we explore how the analytical framework presented in the previous section can be adapted to optimize routing performance metrics in a DTN.

Consider a DTN with m nodes. Each of them implements its own forwarding policy that can be different from node to node, e.g., a node could implement a K-hop scheme, i.e., it would not forward a message that has already traveled more than K hops, while another one could implement a probabilistic scheme, where each message is forwarded with probability p. Even nodes applying the same kind of policy could have different values of the parameters.

Let us denote $\mathbf{z}_i \in \mathbb{R}^{s_i}$ the array of s_i parameters characterizing the policy of node i. Routing performance in the network is determined by the vector $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2 \cdots, \mathbf{z}_m) \in \mathbb{R}^n$, where $n = \sum_{i=1}^m s_i$, which we refer to as the *system status*. It will prove useful to simply identify the component of a vector in $\mathbb{R}^n$ which corresponds to the parameters of node i's policy. Hence, we introduce the notation $[\mathbf{x}]_i$ to refer to the following subvector of $\mathbf{x}$: $(x_{l_i+1}, x_{l_i+2}, \cdots, x_{l_i+s_i})$, with $l_i = \sum_{j=1}^{i-1} s_j$. Using this notation, we have $[\mathbf{z}]_i = \mathbf{z}_i$. We consider that node i can change its parameters at runtime in order to optimize a global performance metric. For simplicity, we assume that such changes occur synchronously at all nodes every T seconds, so that the system can be modeled as a discrete-time system where the slot length is T. Let $\mathbf{z}(k)$ denote the set of parameters used during the time slot k.

We next show that there is a large class of interesting performance metrics that can be mapped to the problem (1). Most of the performance metrics are related to nodes (e.g., energy consumption at each node) or to messages (e.g., delivery time, delivery probability or energy consumption per message). In either case, the performance metrics can naturally be expressed as a sum of local cost functions relative to each node. Let us develop a specific example (that we will consider in Section 4 as a case study). Our target is to minimize a weighted sum of the message delivery time T_D and the number of copies C done for each message, the latter number being roughly proportional to the energy consumption needed to deliver the message. Such quantities are random variables because they depend on the underlying node random mobilities. Hence, we can define our performance metric as :

$$F(\mathbf{z}) = \mathrm{E}[T_D + \gamma C], \tag{4}$$

where the parameter γ can be interpreted as the time-equivalent cost of a copy, and we put in evidence that F depends on all the policy parameters of all the nodes in the network (it depends also on the traffic matrix, but we assume that it is given). The optimization problem we are interested in is to find a parameter vector $\mathbf{z}$ that minimizes F. We observe that (4) can be rewritten as

$$F(\mathbf{z}) = \sum_{i=1}^{m} \mathrm{E}\left[\frac{\lambda_i}{\lambda_T} T_{D,i} + \frac{\mu_i}{\mu_T}\gamma C_i\right], \tag{5}$$

where $T_{D,i}$ is the expected delivery time of a packet generated at node i, C_i is the expected number of copies done by node i for a generic message (generated or not at node i), λ_i is the message generation rate at node i (message for which node i is the source), μ_i is the message arrival rate at node i (accounting for messages generated at or forwarded to node i), $\lambda_T = \sum_{i=1}^{m} \lambda_i$ and $\mu_T = \sum_{i=1}^{m} \mu_i$. According to (5), F is the sum of functions f_i, each of which is relative to a specific node :

$$f_i(\mathbf{z}) = \mathrm{E}\left[\frac{\lambda_i}{\lambda_T} T_{D,i} + \frac{\mu_i}{\mu_T}\gamma C_i\right]. \tag{6}$$

Even if the optimization problem looks similar to the one stated in (1), there are two capital differences :

1. f_i is not in general convex,
2. a closed form expression of f_i (nor of ∇f_i) is not in general available at node i.

Convexity in problem (1) guarantees the absence of local minima. Having non-convex f_i no longer assures that ; nevertheless, the algorithm would in any case converge to a local minimum that may have acceptable performance.

The second point requires a more thorough discussion. Node i cannot evaluate f_i at a generic point $\mathbf{z}$, but, at the end of the k-th time slot, it can *estimate* $f_i(\mathbf{z}(k))$ where $\mathbf{z}(k)$ is the set of policy parameters *used* by nodes during time slot k. [2] For instance, in order to have an estimate of $E[T_{D,i}]$, node i can average the delivery times of the messages it has generated during the slot. This means that a possibly noisy estimate of $f_i(\mathbf{z}(k))$ can be available. Evaluating the gradient is more complex. We believe that derivative-free optimization techniques [13] can be introduced in order to rely only on the estimations of f_i at each node. We do not address this issue in this work, but assume from now on that the gradient of f_i at a given point can be evaluated.

Relying on measurements has another implication for the original algorithm (2). In fact, according to (2), each node should be able to evaluate the gradient in a solution estimate $\hat{\mathbf{x}}^i(k)$ that is potentially different for each node. On the contrary, in a real DTN setting, every node may only have an estimation of its own function f_i evaluated at the same point $\mathbf{z}(k)$. Estimating the value of f_i (or of ∇f_i) at different points appears difficult. We discuss more this issue in the next paragraph.

Let us consider what would an immediate implementation of (2) be, referring to the following set of equations :

$$\mathbf{z}_i(k) = [\hat{\mathbf{x}}^i(k)]_i, \tag{7a}$$

$$\hat{\mathbf{x}}^i(k+1) = \sum_{j=1}^{m} a_j^i(k)\hat{\mathbf{x}}^j(k) - \zeta \nabla f_i(\hat{\mathbf{z}}^i(k)). \tag{7b}$$

2. In reality, as we discuss in the following of this section, node i does not know in general the system status $\mathbf{z}(k)$, but it has an estimate $\hat{\mathbf{z}}^i(k)$.

At the end of the $(k-1)$-st time slot, node i has available a new estimate of an optimal solution $\hat{\mathbf{x}}^i(k)$. It can extract from this estimate the set of parameters $\mathbf{z}_i(k)$ to use during the k-th time slot (cf. (7a)). We observe that the system status is $\mathbf{z}(k) = (\mathbf{z}_1(k), \mathbf{z}_2(k), \cdots, \mathbf{z}_m(k))$ and it is unknown to node i, but for the component $\mathbf{z}_i(k)$. During the k-th time slot node i spreads its current estimate $\hat{\mathbf{x}}^i(k)$ and collects other nodes' estimates. If node i has collected all estimates $\hat{\mathbf{x}}^j(k)$, with $j \neq i$, it can exactly reconstruct $\mathbf{z}(k)$, but in general only a subset of estimates is available so that it can only produce an estimate of system status denoted $\hat{\mathbf{z}}^i(k)$.[3] Once such estimate has been obtained, the gradient ∇f_i can be evaluated at $\hat{\mathbf{z}}^i(k)$ yielding a new estimate of an optimal solution (cf. (7b)).

We note two problems in the above algorithm. First, it is not clear how $\hat{\mathbf{z}}^i(k)$ should be produced. In fact if node i has not met node j, then it has no information about $\mathbf{z}_j(k)$. Second, letting each node select its policy parameters from its own solution estimate $\hat{\mathbf{x}}^i(k)$ leads to a *biased* system status. In fact, ∇f_i contributes to changing all the components of the vector $\hat{\mathbf{x}}^i(k+1)$ and subvectors of ∇f_i relative to node i's policy and to other nodes' policies have different directions in general.[4] However, only the i-th component of $\hat{\mathbf{x}}^i(k+1)$ will effectively be used (by node i); thus, ∇f_i only affects $[\mathbf{z}(k+1)]_i$.

In order to solve these two issues we have changed the algorithm (7) as follows :

$$\hat{\mathbf{x}}^i(k+1) = \hat{\mathbf{z}}^i(k) - \zeta \nabla f_i(\hat{\mathbf{z}}^i(k)), \tag{8a}$$

$$\hat{\mathbf{z}}^i(k+1) = \sum_{j=1}^{m} a_j^i(k) \hat{\mathbf{x}}^j(k+1), \tag{8b}$$

$$\mathbf{z}_i(k+1) = [\hat{\mathbf{z}}^i(k+1)]_i. \tag{8c}$$

In this new procedure a time slot is split into two parts. Since the beginning of the k-th time slot the node has an estimate of the current system status $\hat{\mathbf{z}}^i(k)$. During the first part of the time slot, it estimates $\nabla f_i(\hat{\mathbf{z}}^i(k))$. At the end of the first part, it will produce a new estimate of the optimal solution by correcting $\hat{\mathbf{z}}^i(k)$, taking into account only the gradient ∇f_i (cf. (8a)). The second part of the time slot is then devoted to spreading such estimate and collecting other nodes' estimates. Then, an estimate of the system status in slot $k+1$, $\hat{\mathbf{z}}^i(k+1)$, is obtained by averaging optimal solution estimates (cf. (8b)). This estimate determines also the policy parameters used by node i in slot $k+1$ (cf. (8c)).

We observe that the accuracy of status estimation $\hat{\mathbf{z}}^i(k)$ increases as nodes communication opportunities increase. In particular, if every node can collect estimates of every other node in the second part of time slot k, then $a_j^i(k) = 1/m$ for any i and j and it follows that $\hat{\mathbf{z}}^i(k+1) = \mathbf{z}(k+1)$ for any i. Hence all nodes have the same estimate of the vector of parameters used in the system during the slot $k+1$. We believe results similar to those proved in [12] for (2), and in particular bounds like (3), also hold for our variant (8), but we currently do not

3. Note the difference between $\hat{\mathbf{z}}^i(k)$, the estimate of status $\mathbf{z}(k)$ produced by node i, and $\mathbf{z}_i(k)$, the vector of parameters used by node i and a subvector of $\mathbf{z}(k)$.

4. For example, in a completely symmetric scenario, nodes will have the same type of policy and any two nodes can exchange their policy vectors without affecting the total cost F. More formally, $F(\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_m)$ is invariant to permutations of $\mathbf{z}_i$. For $\mathbf{x}^* \in S^*$, we naturally have $\nabla F(\mathbf{x}^*) = 0$ and this implies, due to the symmetry, that $[\nabla f_i(\mathbf{x}^*)]_i = -(m-1)[\nabla f_i(\mathbf{x}^*)]_j$ for $j \neq i$. The corrections introduced by f_i have opposite directions.

set	α	β	γ	δ	ϵ	ζ	λ_i	m
A	0.01	1	10	1	0.1	$3.35e^{-5}$	$1\,\forall i$	20
B	0.02	1	10	1	0.1	$10^{-4} \rightarrow 0.5$	$1\,\forall i$	10

TABLE 1 – Parameters settings

have a proof. For the particular case when all estimates are available at every nodes in every time slot, then it can be proved that (8) reduces to a distributed implementation of a classic gradient method, so that all standard convergence results apply. We will refer to this case as the *global knowledge scenario*.

4 A Case Study

In this section we perform a preliminary numerical evaluation of the algorithm (8) for cost function $F(x) = \mathrm{E}[T_D + \gamma C]$. In particular, we are interested to investigate if the algorithm converges, how much time is required and how good is the final selected operation point.

We consider a DTN with m mobile nodes, all of them implementing probabilistic forwarding with a different probability value : $p_i(k)$ is the value used by node i during slot k. The system status vector during the k-th slot is $\mathbf{z} = \mathbf{p} = (p_1(k), p_2(k), \cdots p_m(k)) \in \mathbb{R}^m$. We assume a global knowledge scenario.

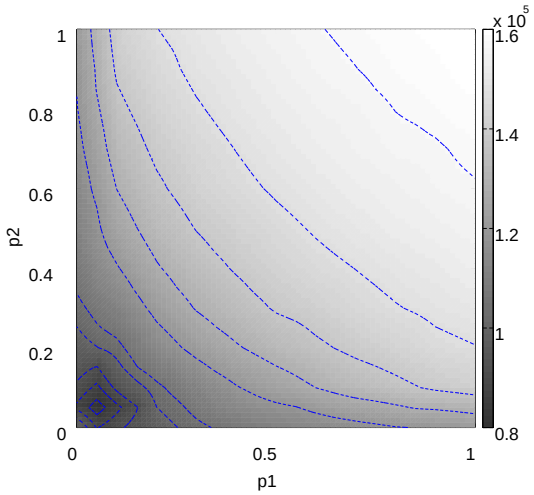
For our purpose we have considered the following approximation for the local cost function $f_i()$:

$$f_i(\mathbf{p}) = \frac{\lambda_i}{p_i + \alpha \sum_{j \neq i} p_j + \iota} + \gamma \left(1 - e^{-\beta p_i}\right) \left(\lambda_i \delta + \epsilon \overline{\lambda}_{-i}\right), \tag{9}$$

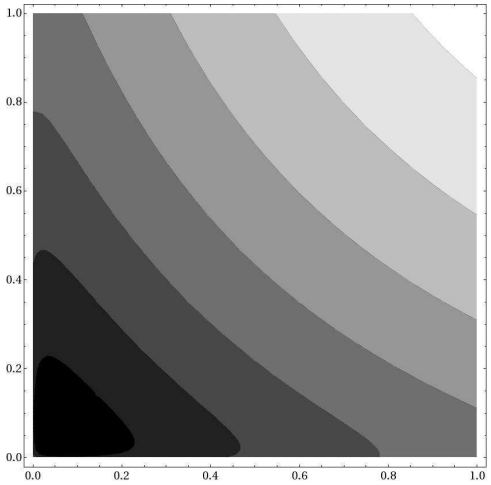
where $\overline{\lambda}_{-i} = \sum_{j \neq i} \lambda_j$. In fact, for opportune values of the parameters α, β, γ, ϵ and ι, this function can produce the same qualitative behaviour of our cost function. For example Fig. 1 shows contour plots for the total cost $F = \sum f_i$ for a network with 100 nodes, where half of them adopt probability $p1$ and the other adopt probability $p2$. Values in Fig. 1(a) have been obtained by simulations [5], while those in Fig. 1(b) have been obtained through (9) with an appropriate choice of the parameters. Unless otherwise specified, results in this report have been obtained for one of the two sets of parameters specified in Table 4. The ζ value in set A has been selected inversely proportional to the maximum gradient of the function $F(p)$. This choice prevents algorithm oscillations. In set B we explore a large range of values for parameter ζ.

Fig. 2 shows a temporal evolution of the cost function for the parameter set A. After 10^5 iterations $p(k)$ converges to a homogeneous vector ($p_i \approx 0.3241\ \forall i$) and the value of the total cost is $F \approx 112.6$. We conducted different experiments changing the initial vector $p(0)$ but the algorithm has always converged to the same value. Moreover by sampling directly the function $F(p)$ it seems that the algorithm has correctly identified the global minimum.

5. The complete description of the network setting (mobility model, network area, transmission range, message generation process ...) can be found in Section 4.2 of [11]. For the purpose of such qualitative comparison, we thought it was not needed to provide a complete description.



(a) Simulation results



(b) Approximated results

FIGURE 1 – Total cost $F(p)$ when half of the nodes adopt probability $p1$ and half adopt probability $p2$

For different values of α, e.g. for $\alpha = 0.1$, the function has multiple local minima, in particular the set of global minima seems to be the set of all the heterogeneous vector p, where 5 components have value equal to 1 and the others equal to zero. Most of the time the algorithm has been converging to one of these equivalent configurations. In a few cases it has converged to a different local minimum.

The order of magnitude of the convergence time in the above example is around 10^5 iterations, that is probably unacceptable in a real implementation. In fact, during each iteration, a node has to estimate ∇f_i for the current setting and this can require the delivery of many messages in order to have acceptable

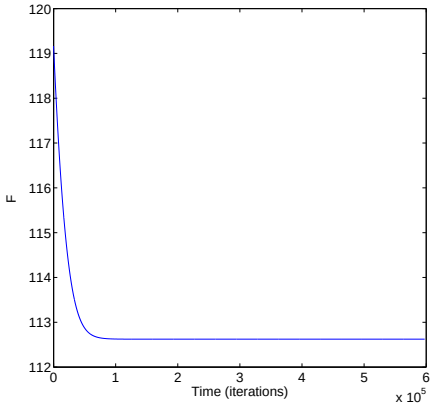


FIGURE 2 – Convergence time plot

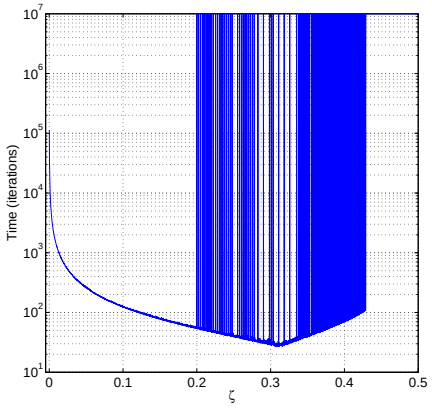
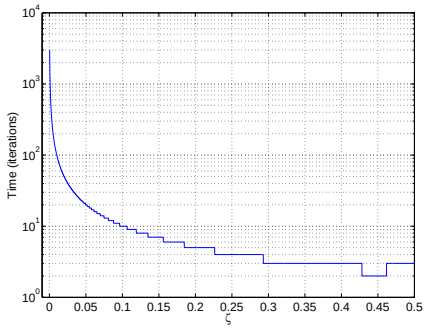


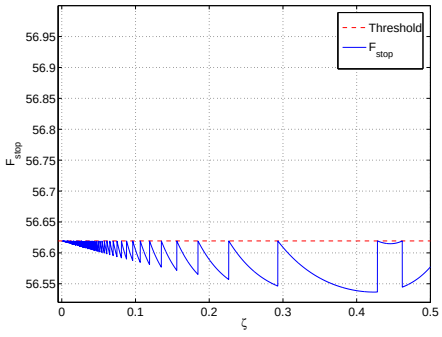
FIGURE 3 – The effect of the parameter ζ on convergence time.

levels of estimation noise. For this reason we have explored the behaviour of the algorithm for larger values of ζ, that should allow faster convergence.

Figure 3 shows the convergence time starting from a given initial system status $\mathbf{z}(0)$ for different values of the parameter ζ. The convergence time is evaluated as the number of iterations of (8) until the value of each component of $\hat{\mathbf{z}}(k)$ does not change more than 10^{-15} (Matlab floating point precision). Whenever convergence has been reached, the final value of $F(\mathbf{z})$ -say it F_{stop}- has been equal to 56.53, which seems to be the global minimum of the function (again by sampling directly the function F). Spikes for $\zeta > 0.2$ correspond to ζ values for which the algorithm does not converge by 10^7 iterations (we stop calculations when such iteration number is reached). This is the case for all the values larger than 0.43. Analyzing some of these time series, it appears that $\mathbf{z}(k)$ reaches a periodic orbit. In any case, if we average the last 10 values of $F(\mathbf{z})$ we obtain again 56.53. This means that such periodic orbits are very close to



(a) Convergence Time



(b) Cost after convergence

FIGURE 4 – Convergence to an interval

the global minimum. Hence, from a practical point of view, the algorithm has identified the same minimum for all the tested values.

It is interesting to observe the behaviour of the convergence time if we filter away the spikes. As ζ increases, the convergence time first decreases and then increases (for $\zeta > 0.31$). This can be explained as follows. There are two phases that can be in general identified in a sequence $\mathbf{z}(k)$ that converges to $\mathbf{z}^*$. In the first phase, $\mathbf{z}(k)$ moves more or less in the same direction approaching $\mathbf{z}^*$, then it start bouncing around $\mathbf{z}^*$ until it does not fall in the interval of values that are considered equivalent to $\mathbf{z}^*$. As ζ increases, the length of the first phase decreases (less jumps are needed to arrive close to $\mathbf{z}^*$), but, on the other side the length of the second phase increases (more jumps are needed to fall exactly in the interval). Finally, we observe that under the best setting, only 30 iterations are needed to reach the optimal configuration.

The convergence time can be further reduced if we sacrifice the accuracy of the solution. In Figure 4 the algorithm is stopped once $F(\mathbf{z}(k))$ falls below $\tilde{F} = 56.62$. This value has been calculated as $\tilde{F} = F^* + 0.01|\overline{F} - F^*|$, where $F^* = 56.3$ is the minimum of the function and $\overline{F}$ is its average value [6]. Figure 4(a) shows a reduction of one order of magnitude of the convergence time ; in some cases

6. Both values have been estimated by sampling the function.

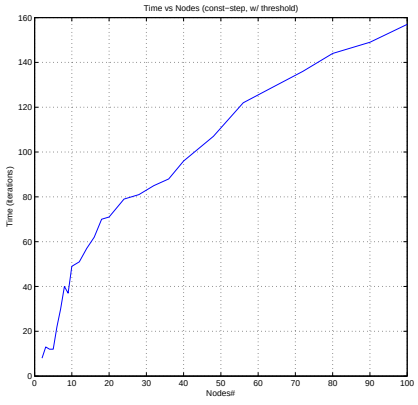


FIGURE 5 – Convergence time vs nodes number

only a couple of iterations. Figure 4(b) shows the corresponding loss of accuracy (remember that the minimum is $F^* = 56.3$).

Finally, we wanted to study how the convergence time scales with the number of nodes m. Being that it is not easy to select the optimal value of ζ for each m and performance are very sensitive to such value, we have considered a variant of (8a), where at each step the correction to the status vector has a constant module equal to ϕ. This is achieved by replacing Eq. (8a) with

$$\hat{\mathbf{x}}^i(k+1) \quad = \quad \hat{\mathbf{z}}^i(k) - \phi \frac{\nabla f_i(\hat{\mathbf{z}}^i(k))}{||\sum_j \nabla f_j(\hat{\mathbf{z}}^j(k))||}. \tag{10a}$$

Figure 5 shows the convergence time until $F(\mathbf{z}(k)) < \tilde{F}$ for $\phi = 0.01$ [7]. Interestingly, the convergence time appears to scale less then linearly with the number of nodes.

5 Conclusions

In this report we have proposed to use the new analytical framework for multi-agent optimization proposed in [12] in order to optimize routing in Delay Tolerant Networks. In Section 3, we have pointed out some issues to be addressed in order to apply this framework in a realistic case where no analytical expression is available for the function to optimize nor for its derivatives, but estimates can be obtained from measurements. We have changed the original algorithm in order to partially addresses these issues, but further research is needed in order to complement the framework with derivative-free optimization techniques. The preliminary evaluation in Section 4 shows promising results. The convergence of the algorithm does not seem to be significantly dependent on the gradient step size ζ and a careful tuning of such value allows to achieve practically useful convergence time values. Finally, the algorithm appears to scale well with the number of nodes.

7. Note that $\tilde{F}$ has to be evaluated separately for each m. In fact the function F depends on m, so that both minimum and average values changes as m change.

Références

[1] S. Burleigh, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, "Delay-tolerant networking : an approach to interplanetary internet," *IEEE Comm. Mag.*, vol. 41, no. 6, pp. 128–136, Jun. 2003.

[2] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic networking : data forwarding in disconnected mobile ad hoc networks," *IEEE Comm. Mag.*, vol. 44, no. 11, pp. 134–141, 2006.

[3] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Efficient routing in intermittently connected mobile networks : The multi-copy case," *ACM/IEEE Trans. on Networking*, vol. 16, pp. 77–90, Feb. 2008.

[4] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke University, Tech. Rep. CS-2000-06, 2000.

[5] R. Groenevelt, P. Nain, and G. Koole, "Message delay in mobile ad hoc networks," *Perf. Eval.*, vol. 62, no. 1-4, pp. 210–228, October 5-7 2005, proc. of Performance 2005, Juan-les-Pins, France.

[6] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," in *Proc. of SAPIR Workshop*, ser. LNCS, vol. 3126, 2004, pp. 239–254.

[7] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait : an efficient routing scheme for intermittently connected mobile networks," in *Proc. of ACM WDTN*, 2005.

[8] G. Neglia and X. Zhang, "Optimal delay-power tradeoff in sparse delay tolerant networks : a preliminary study," in *Proc. of ACM SIGCOMM CHANTS*, 2006, pp. 237–244.

[9] E. Altman, G. Neglia, F. D. Pellegrini, and D. Miorandi, "Decentralized stochastic control of delay tolerant networks," in *INFOCOM*, 2009.

[10] S. Alouf, I. Carreras, D. Miorandi, and G. Neglia, "Embedding evolution in Epidemic-Style forwarding," in *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, 2007, pp. 1–6.

[11] S. Alouf, I. Carreras, A. Fialho, D. Miorandi, and G. Neglia, "Autonomic information diffusion in intermittently connected networks," in *Autonomic Computing and Networking*, M. Denko, L. Yang, and Y. Zhang, Eds. Springer Verlag, June 2009, pp. 411–433.

[12] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for Multi-Agent optimization," *Automatic Control, IEEE Transactions on*, vol. 54, no. 1, pp. 48–61, 2009.

[13] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics, 2009.

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399