Distributed Stochastic Optimization in Opportunistic Networks: The Case of Optimal Relay Selection

Andreea Picu
ETH Zürich
Communication Systems Group
Zürich, Switzerland
picu@tik.ee.ethz.ch

Thrasyvoulos Spyropoulos
ETH Zürich
Communication Systems Group
Zürich, Switzerland
spyropoulos@tik.ee.ethz.ch

ABSTRACT

Opportunistic Networking allows wireless nodes to exchange data and information of interest with peers in communication range. These nodes form a large, dynamic, multi-hop network "on the fly". Challenging optimization problems arise, such as end-to-end routing, resource allocation (e.g., for buffer space and bandwidth), content placement etc., exacerbated by the lack of end-to-end connectivity. While globally optimal solutions are normally sought in network optimization, node actions and decisions in this context are inherently *local*. As a result, most solutions proposed rely on local heuristics without any guarantees about their convergence properties towards a desired global outcome. In this paper, we argue that the framework of Markov Chain Monte Carlo (MCMC) optimization can be applied to many problems in Opportunistic Networking, providing efficient *local algorithms that provably converge to a globally optimal solution*. As a case study, we use the problem of optimal relay selection for group communication (e.g., multicast), based on node contact patterns.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design—*Store and forward networks, Wireless communication*; C.2.2 [**Computer Communication Networks**]: Network Protocols—*Routing Protocols*

General Terms

Algorithms, Design, Performance, Theory, Verification

Keywords

DTN, Distributed Optimization, Markov Chain Monte Carlo, Content Placement, Contact Graph

1. INTRODUCTION

With the increasing integration of wireless short and medium range communication technologies (Bluetooth, 802.11/WiFi) into mobile devices, the formation of a large, dynamic, multihop network "on the fly" is possible. This type of network, also known as an *Opportunistic Network*, can enable novel applications based on spontaneous communication, interaction, and collaboration (e.g., social networking, location-based services, micro-blogging etc.), as well as support traditional ones, such as file sharing, chatting, and web browsing.

Connectivity in Opportunistic Networks is usually intermittent and partial (i.e., not end-to-end) and the *contact* (temporary physical proximity of two devices) is the only means to spread data. Common network optimization problems such as routing, congestion control, and resource allocation, whose distributed solution is already challenging, are further complicated in this context. Here are some problems often addressed in Opportunistic Networking research:

- *Routing.* A node carrying messages must make local decisions about which messages to forward upon contact with other nodes [19, 23, 5, 12]. Locally, the goal is to improve the delivery probability with each decision. Globally, the goal is for individual decisions of nodes to optimize performance (e.g., maximize delivery ratio) for all messages.
- *Buffer Management.* When node buffer capacity is limited, each node must locally decide which message to drop [1, 16]. Globally, the goal is once more to maximize performance for all messages in the network.
- *Content/service placement.* New content is injected into the network, in which a large subset of nodes (e.g., a multicast group) might be interested over time [17]. To make the content easily reachable by interested nodes, L replicas are pushed from its source to L "carriers", which will make it most available to everyone (e.g., minimizing the expected meeting time of an interested node and a carrier). The optimal configuration (i.e., set of L carriers) requires global knowledge [21]. Yet, each node must locally decide who to forward the replica to, to reach the optimal configuration, without explicit knowledge about other replicas.

It is evident that globally optimal solutions are sought in the above network optimization examples. However, node actions and decisions in this context are inherently *local* without any end-to-end semantics or global knowledge about the network. In fact, limited and frequently changing connectivity implies that up-to-date knowledge cannot ever be obtained, unlike in more traditional distributed problems. As a result, most solutions proposed rely on local best effort heuristics without any guarantees about their convergence properties towards a desired global outcome.

To this end, in this paper we take a first look into the general problem of distributed optimization for Opportunistic

CHANTS'10, September 24, 2010, Chicago, Illinois, USA.

Networks. We examine two types of algorithms: (i) greedy algorithms, corresponding to gradient-ascent based optimization and (ii) stochastic optimization, based on Markov Chain Monte Carlo methods. We argue that the framework of Markov Chain Monte Carlo (MCMC) optimization can be applied to many problems in Opportunistic Networking, providing efficient *local algorithms that provably converge to a globally optimal solution*.

Throughout this work, we use the optimal relay selection problem, described above, as a case study. This is an open problem when considering realistic heterogeneous contact patterns between nodes, with complex "social" structure, as discussed in [12]. While most of the research in Opportunistic Networks has focused on unicast routing, we believe that group communication (e.g., multicast, anycast) and publish/subscribe models will be equally, if not more, relevant to applications envisioned for these networks. Hence, a potential solution to this problem would be of general interest.

We identify different cost functions for this optimal relay selection problem and evaluate both greedy and stochastic gradient search algorithms on real and synthetic traces. Our findings suggest that MCMC-based algorithms are able to efficiently navigate through local maxima, when these exist, and converge to desirable configurations. Surprisingly, simple greedy algorithms are often able to achieve good performance, in this context. We conjecture that this property is related to the dynamicity of the "local" neighborhood for gradient-based searching, in contrast to the static neighborhood of traditional cost functions. We believe these preliminary findings are more generally applicable to the problem of distributed optimization in Opportunistic Networks.

The remainder of this paper is organized as follows: In Section 2, we review previous studies concerning global optimization in contexts similar to ours. Section 3 formally introduces our network model and the various cost/utility functions we use. The two algorithms under consideration are described in Section 4. We show our empirical evaluation of the algorithms in Section 5. Finally, we present our future work plans and conclusions in Section 6.

2. RELATED WORK

Numerous aspects of networked or distributed systems are configurable or depend on parameter settings, such as topology, routing, congestion control etc. Network optimization attempts to find optimal configurations of different features of distributed systems. In [24], Weise et al. provide a thorough review of a number of optimization-related works in the area of distributed systems. Techniques like Evolutionary Algorithms, Ant Colony Optimization, Simulated Annealing and Tabu Search are discussed, along with their application to problems as diverse as network topologies, generating routing protocols, security, and software configuration.

In the area of Opportunistic Networking, as mentioned earlier, a number of these optimization problems arise, yet are usually treated using heuristics. One of the first works to formulate a DTN-related problem, namely end-to-end routing, expressly as an optimization problem is [1]. In this work, an *intentional* routing protocol is proposed, which attempts to explicitly optimize user-specified routing metrics. The authors prove that under the considered conditions, the routing problem is NP-hard; they propose instead that each node implements a greedy algorithm that chooses locally between messages (e.g., to forward) according to a per message utility. Yet, no guarantees are given about the performance of this algorithm relative to an optimal one.

More recently, Hu et al. have proposed in [11], to use a Markov Chain Monte Carlo algorithm for the optimization of the dissemination time in collaborative ad hoc information dissemination. In this application, devices help in forwarding information channels to the entire network, by disseminating the channels they subscribe to, plus others. [11] treats the case where devices have limited storage, and thus must decide which channels they are willing to help disseminate. The authors use a Markov Chain Monte Carlo algorithm to find channel selection strategies which provably optimize the dissemination time across the channels.

Markov Chain Monte Carlo algorithms have also been applied in an area more closely related to Opportunistic Networking: Mobile Ad Hoc Networking. Kauffmann et al. apply an MCMC algorithm to the problem of distributed channel selection in MANETs, in [14]. Yu et al. use an MCMC method, in [25], to solve a resource replication problem. Their end goal is to optimize for application requirements such as: load balance, connectivity, energy etc.

Algorithms using the Markov Chain Monte Carlo framework are also relatively popular in other networking contexts. In [22], Sandberg describes an MCMC-based routing algorithm for peer-to-peer networks, which is now used in the Freenet Project. In sensor networks, [13] uses an MCMC scheme for distributed estimation.

Our main contributions in this paper are the following: (i) we address the problem of content placement and group communication in networks with largely heterogeneous mobility patterns as a distributed optimization problem, (ii) we derive a simple distributed solution for this problem, based on Markov Chain Monte Carlo optimization, and demonstrate the general applicability of the MCMC framework for solving difficult Opportunistic Networking problems.

3. PROBLEM AND NETWORK MODEL

Many problems arising in the context of Opportunistic Networks can be modeled as *combinatorial optimization* problems. This comes naturally, as we are dealing with indivisible entities (nodes, messages, channels etc) and with rules that define a finite number of allowable choices (choice of relays, assignment of channels etc). A more formal definition of a combinatorial optimization problem in this context is:

Definition 1 (Combinatorial Optimization Problem). *A combinatorial optimization problem, $\mathcal{O}$, is defined by a set of allowable configurations (e.g., subset of nodes, assignment of objects to nodes), given a set of constraints, and a cost or utility function $U_{\mathcal{O}}$. Given a configuration s, the utility function outputs either $U_{\mathcal{O}}(s)$, the cost of the solution (a non-negative real number), or the special value $\perp$, if s is not a feasible solution for $\mathcal{O}$ (does not meet the constraints). The goal is to find a feasible solution s, such that $U_{\mathcal{O}}(s)$ is minimized (for cost) or maximized (for utility).*

In general, there are two major challenges when considering such optimization problems in the context of (Opportunistic) networks. First, obtaining a solution to many interesting combinatorial problems is often NP-hard. Thus, finding a direct solution is often prohibitive for small wireless devices. Second, calculating $U_{\mathcal{O}}(s)$ often requires central knowledge. As a result, optimal centralized algorithms

do not always lend themselves to optimal distributed implementations. Furthermore, obtaining up-to-date global knowledge is usually infeasible in Opportunistic Networks due to connectivity considerations and high node churn.

To address the issue of complexity, stochastic optimization and approximation algorithms are usually applied, yet in the context of centralized solutions. To enable distributed implementations of general network optimization problems, Chen et al. propose a systematic transformation of utility functions using log-sum-exp functions [4]. While this transformation yields an *approximation* to the original problem, it also enables a distributed solution. To cope with the issue of global knowledge, estimates of required global parameters can be used [11, 16].

3.1 Network Model

We will focus here on optimization for Opportunistic Networks. The general network model considered is as follows. Let $\mathcal{N}$ be the set of all nodes in the Opportunistic Network under consideration, $|\mathcal{N}| = N$. Each of the N nodes is identified by a unique ID and its mobility is assumed to be governed by (implicit or explicit) social relations. Specifically, (i) we can identify *node communities*, i.e., sets of nodes that tend to meet each other preferentially, and ii) nodes have different *sociability* or number of nodes they meet in a given time interval, ranging from *solitary* to *gregarious*. This seems a reasonable assumption, since mobile devices (network nodes) are carried by humans, who engage in socially meaningful relationships. Several previous DTN experiments, [6, 18], have confirmed this type of interaction patterns and information flow. We assume a *social graph* is created using past *contacts* between nodes.

Contacts. A *contact* between two nodes happens when those nodes setup a bi-directional wireless link between them. We assume contacts last for a negligible time compared to that between two successive contacts, but long enough to allow all the required exchanges to happen. For the purpose of analysis, we also assume that contacts occur in sequence, i.e., the probability of simultaneous contacts is small[1].

Social graph. A *social graph* represents our network: nodes (mobile devices) are vertices and existence of an edge implies a "strong" relationship (e.g., a recent contact, or frequent contacts) between the two endpoints. The graph seeks to capture the aforesaid social features of this network. Ways of creating the social graph of a DTN, out of a sequence of occurring contacts are implicitly used in various previous works [5, 12]. More recently, [9] explicitly addresses this as a standalone issue. Here, we build the social graph using the density-based aggregation described there.

3.2 Optimal Relay Selection

Using the above network model, we define here the problem of optimal relay selection given a fixed budget of message/service replicas.

Definition 2 (Optimal Relay Selection). *A source node $n_s \in \mathcal{N}$, creates a message/service m that can be consumed by any interested node in the network. To maximize the availability of m to requesting nodes, the source must find a subset of nodes $\mathcal{L}$ of size $L = |\mathcal{L}|$, which will permanently (or while m is valid) store a copy of m. The optimization goal can be to (i) minimize the expected discovery/delivery delay for any requesting node, or (ii) maximize the number of requests served before a deadline (time to live (TTL)) .*

[1]This is the case, for example, if we assume that the arrival process of contact events is Poisson. In general, this assumption just implies a relatively sparse network.

The assumption of a limited budget of replicas is reasonable if we consider a large, resource-constrained Opportunistic Network. A vast amount of user-generated content is expected to be shared over it, each content possibly of large size (e.g., music files, videos, images etc.). As a result, nodes will not be able to locally store all available content. While different flavors of the problem can be conceived (e.g., minimizing the number of replicas given a performance constraint), their solution could be cast in a similar framework.

The above problem consists of the following two subproblems. First, given the entire contact graph, choose the L relays that would minimize the expected content discovery time for any node. This essentially depends on the mobility properties of the nodes in the network. Since the contact graph captures node mobility properties, the choice of relays depends on the relative "position" of candidate relays in this graph. Nevertheless, knowledge of the entire graph is not available locally, and local "importance" metrics such as degree centrality and ego-centrality are usually considered [5, 12]. In [21], we have proven that, if one considers degree centrality only (i.e., the number of neighbors of each node), the optimal relay set $\mathcal{L}$ consists of the highest degree nodes.

Having identified the optimal set of relays, the second problem consists in actually "pushing" the L replicas from a source to the chosen relays. This non-trivial problem is the optimization case study we consider here. There are two major challenges in this context. First, as mentioned earlier, a source node does not have a global picture of the network, and thus cannot know a priori the optimal set of relays (e.g., the highest degree nodes, as defined in [21]). Consequently, a set of local forwarding rules is required that can deliver the replicas to the optimal relays *without prior knowledge about them*. Second, complex mobility patterns of nodes involved have an important impact on both the *convergence* time (i.e., time to distribute all copies to the optimal relays) and the quality of "found solution", as some replicas may get stuck or oscillate due to local maxima.

To address these issues we will take the following approach. Once a node creates a message, it binary sprays the L copies to the first nodes it meets [23]. This quickly creates an initial configuration of L copies at L (random) nodes. This is also the starting point of the optimization algorithm. The goal is, from the initial configuration, to reach the optimal configuration by gradually improving the set of relays. However, in order to evaluate the quality of a solution $\mathcal{L}$, a utility or cost function is needed. Some candidate utility functions are examined in the following.

3.3 Utility Functions

As discussed in the previous section, it has been proved in [21], that the nodes of maximum degree are the optimum relays, when degrees are the only information available to nodes. In this case, the best utility function to maximize is the sum of the degrees of the relays.

When more information is assumed available, various other utility functions are possible. While node degrees express node sociability (and are important in scenarios with skewed degree distributions), node communities are another salient characteristic of social mobility. Covering all communities would be an additional goal of the relay selection algorithm.

Yet, this requirement is sometimes in conflict with the degree maximization requirement, as high degree nodes might reside in the same community. To capture this, one could assume that each node also has knowledge of the identities of its neighbors, not only their number. In this case, an intuitive utility function for minimizing the delivery time would be the total number of nodes "covered" by the set of relays or having at least one neighbor that is a relay[2].

More formally, these utility functions could be written as:

$$U_1(m,\mathcal{N},\mathcal{L}) = \sum_{1\leqslant i\leqslant L} d_i \qquad \text{(degree only)}, \quad (1)$$

$$U_2(m,\mathcal{N},\mathcal{L}) = \left|\bigcup_{1\leqslant i\leqslant L} \Gamma(n_i)\right| \qquad \text{(degree and community)}, \quad (2)$$

with $\Gamma(n_x)$, the set of neighbors of node n_x and $|\Gamma(n_x)| = d_x$.

4. DISTRIBUTED OPTIMIZATION ALGORITHMS

We have thus far described the network model assumed, the network (combinatorial) optimization problem(s) we are interested in solving in this context, and appropriate utility functions to evaluate the quality of different configurations (i.e., solutions). Our starting point is the initial configuration reached after the binary spraying phase. In this section, we will present two types of algorithms to progress from *any* initial configuration to the globally optimal one.

When the optimization (utility) function is convex, gradient-ascent algorithms can be used [2]. Since the solutions space for our problem is discrete, this would correspond to the following rule: define a *neighborhood* around the current configuration, and replace the current configuration with the one (within the neighborhood) that most improves the utility function. The choice of neighborhood is a tradeoff between speed and convergence accuracy.

In the context of Opportunistic Networks, the sequence of steps an optimization algorithm goes through is determined by node meetings. At each contact between two nodes, the algorithm is presented with a set of choices, usually messages-related. Examples are forwarding a message, replicating it, or dropping it. The choice made determines the new configuration. In the context of relay selection, the state of only one replica can change. Hence, the local search neighborhood consists of all (or a subset of) configurations with at most one different relay compared to the current one (note this is related to Gibbs sampling [3]). However, because the decision is made at the level of each node, during a contact, a distributed implementation of the gradient-ascent algorithm is presented with only two candidate configurations involving the two nodes in contact. It can evaluate the utility of the two configurations, and choose accordingly.

The first algorithm we consider is a *greedy algorithm*, that deterministically chooses the highest utility configuration at every step. While this converges to an optimal solution if the utility function is convex, its performance suffers in the presence of local maxima. To address this issue we present a stochastic algorithm, based on the Markov Chain Monte Carlo method that introduces a *carefully chosen* amount of randomization to evade local maxima and converge to the global maximum. Both of these algorithms are distributed, as configurations are improved locally, contact-by-contact, with the hope to reach an optimal one.

[2]We note that this problem is related to the *partial dominating set* problem, known to be hard [15]. It is beyond the scope of this paper to find the best (approximation) solution to this problem and a respective utility function. Instead, we are interested in evaluating a gamut of relevant utility functions and their effect on distributed optimization algorithms.

4.1 The Greedy Algorithm

For the optimal relay selection problem defined, the greedy algorithm is applied after the binary spraying stage. Each node taking part in a contact behaves as in Algorithm 1.

Algorithm 1: `Greedy` for Optimal Relay Selection

Data: n_a applies `Greedy` upon meeting n_b. $\mathcal{L}_a$, $\mathcal{L}_b$ the set of relays with n_a, and resp. with n_b.
Result: For each message carried by n_a, potentially a new configuration.

```
1.1 for m ∈ Messages(n_a) do
1.2     U_a ← U_M(m,N,L_a);
1.3     U_b ← U_M(m,N,L_b);
1.4     if U_b ⩾ U_a then
1.5         Messages(n_a) ← Messages(n_a) ∖ m;
1.6         Messages(n_b) ← Messages(n_b) ∪ m;
```

While the search over configurations is performed in a distributed way, the above description implies that, in the general case, global knowledge may be needed to evaluate the utility of each configuration. Indeed, as shown in the beginning of Section 3, nodes n_a and n_b may need to know which $L-1$ other nodes currently hold a copy of message m, in order to calculate utilities U_a and U_b in Algorithm 1. Therefore, additional measures are required to ensure that the utility function can be evaluated or *estimated* locally.

4.1.1 Distributed Utilities

To enable a fully distributed implementation of the optimization algorithm at hand, the utility function should be suitably chosen, such that it can be evaluated locally. This implies that *the marginal contribution of each node (for each state) should be independent of the state of other nodes.* For example, this is the case when the utility function is the sum of the state (e.g., relay or non-relay) or performance (e.g., throughput) of individual nodes (see e.g., [4]). While this is not always possible, it implies that appropriate approximations or local estimates of the utility should be sought, that can be decomposed in such a manner.

Let us consider the utility function U_1 in equation 1. With this function, line 1.4 in Algorithm 1 can be rewritten as $U_b - U_a \geqslant 0$, which reduces to $d_b - d_a \geqslant 0$, according to equation 1. Thus, when using utility U_1, the greedy algorithm forwards messages only to nodes of higher degree than the current.

In the case of utility function U_2, things are more complicated. Community structure introduces inter-dependencies between nodes. From equation 2, U_2 can be written as:

$$U_2(m,\mathcal{N},\mathcal{L}) = \left|\bigcup_{1\leqslant i\leqslant L} \Gamma(n_i)\right| = \sum_{1\leqslant i\leqslant L} d_i - \sum_{k=2}^{L}(-1)^k \sum_{1\leqslant x_1<\ldots<x_k\leqslant L} |\Gamma(n_{x_1})\cap\cdots\cap\Gamma(n_{x_k})|\,. \quad (3)$$

Using the same rewriting as before, for line 1.4 in Algorithm 1, we can obtain a similar result. We denote by

$C^a_{L-1} = \bigcup_{\substack{1 \leqslant i \leqslant L \\ i \neq a}} \Gamma(n_i)$, the coverage of the $(L-1)$-node set of relays formed by excluding the relay node involved in the contact, here n_a. The corresponding utility is $U_2(m, \mathcal{N}, \mathcal{L} \setminus \{n_a\}) = |C^a_{L-1}|$. Then,

$$U_b - U_a = \left[d_b - \left|\Gamma(n_b) \cap C^a_{L-1}\right|\right] - \left[d_a - \left|\Gamma(n_a) \cap C^a_{L-1}\right|\right] = d_b^{\text{eff}} - d_a^{\text{eff}}. \quad (4)$$

In equation 4, we call the two terms of the difference, the *effective degrees* of nodes n_a and, respectively n_b, for the message in question. For each message, the effective degree is defined as the number of nodes that are uniquely covered by a relay, that is, none of the other relays covers them. The above equation still requires knowledge of the other relays and their neighborhoods. Yet, it also suggests a possible way to estimate each node's effective degree locally.

A node can maintain its effective degree (per message), the value of which gets updated upon encounters with other nodes carrying the same message. In that event, the node with the lower degree would subtract the number of neighbors in common from its current effective degree for that message. The node with the higher degree would keep its current effective degree. Through this algorithm nodes inside the same community quickly converge to their effective degree, which is exactly the desirable behavior since overlaps mainly occur between nodes sharing a community. Nevertheless, this simple algorithm does not guarantee an accurate estimate in the worst case. While the issue of distributed estimation is equally important (see e.g., [8]), our focus in this paper is distributed optimization in the presence of local maxima. To this end, we will assume the effective degree is known during any evaluation concerning utility U_2.

4.1.2 Impact of Local Maxima

While deterministic gradient-ascent algorithms (i.e., greedy) do well for convex optimization functions, they usually fail in the presence of complex non-linear utility functions with many local extrema, i.e., points where the utility function is higher than all points inside the local search neighborhood. In the context of Opportunistic Networking and related optimization problems described earlier, it is not fully clear what a local maximum corresponds to. Existence of local extrema heavily depends on the mobility pattern of nodes involved *and* the utility function at hand. Furthermore, the "local neighborhood" changes due to node mobility, and thus a local maximum could be only *temporary*.

To demonstrate our point, let us consider the following community-based mobility scenario, generated with the mobility model of [10]. This model attempts to capture the observed preference of most nodes to move within a small, "home" subset of locations most or all of the time. A simple example instance of this scenario is shown in Figure 1. There are a number of communities in the network (4 in our example): The majority of nodes are *community nodes*. Community nodes only move inside their community. There are also a small subset of *roaming* or *bridging* nodes that move freely around the entire simulation area. Finally, the transmission range is such that nodes from adjacent subareas cannot communicate with each other.

In this class of scenarios, if we consider the aggregated contact graph [9], it is reasonable to assume that we have a set of non-overlapping communities, and that the only means of communication among them is through the bridge

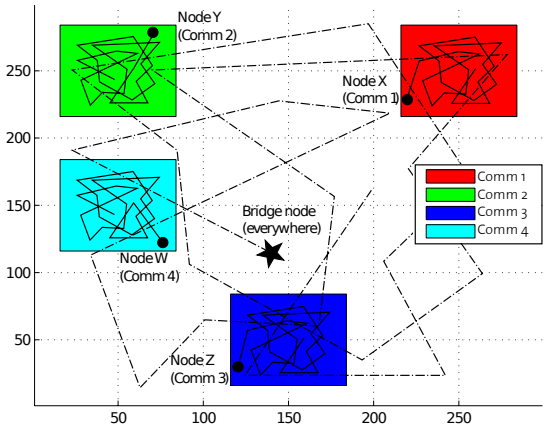


Figure 1: Worst case scenario

nodes. Moreover, if each community consists of a large number of nodes, the degree of community nodes is expected to be higher than the degree of bridge nodes, since the latter roam a much larger area, encountering quite fewer nodes during a time window (this is the case for both "most recent" and "most frequent" aggregation [9]).

Evidently, this creates a problematic situation for the greedy algorithm. Message replicas cannot exit a community, and thus will never reach the optimal set of relays, since bridges have much smaller degrees and will never be chosen by the greedy algorithm. This is clear in the case of U_1, and also the case in most simulated instances for U_2 (see in Section 5).

While it would be of great interest to rigorously define the conditions under which such local extrema arise, as a function of the mobility model and utility function, and to more deeply understand the impact of their existence in the context of Opportunistic Networks, the above scenario clearly demonstrates that a different type of algorithm is needed, that is able to cope with such scenarios. We defer a detailed analytical treatment of this problem for future work.

4.2 Markov Chain Monte Carlo Methods

As shown above, situations arise, when the greedy algorithm will be stuck in a local optimum. To cope with this, stochastic optimization uses randomness in the algorithm, in order to provide the opportunities to move away from a local solution. While randomization is required, we would still like to converge to near optimal solutions. Randomization itself will not suffice, as this would essentially correspond to a *random walk* over the configuration space. In the context of stochastic optimization, this is often achieved via *simulated annealing* techniques, where the amount of randomization decreases over time (more about this later).

In the context of Opportunistic Networking, the type of solution-space-traversal permissible by occurring contacts can be naturally mapped to Markov Chain Monte Carlo (MCMC) methods [3]. While MCMC methods are often used to simulate and sample complex (and non-invertible) functions, they also provide a powerful tool for stochastic optimization. They allow moving to lower utility states, but calibrate the probability of such moves so as to provably converge to an optimal solution [11, 14]. We will consider here the Metropolis-Hastings algorithm.

4.2.1 The Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm is a sampling algo-

rithm. In the context of optimization, it consists of building an *ergodic* Markov chain, whereof the states are feasible solutions of the optimization problem at hand. For the chain to converge to the optimal solution, the stationary distribution of the chain must be concentrated around that solution.

Concretely, when applying a Metropolis-Hastings algorithm, one needs the following:

- *Probability distribution to be sampled*: $\pi(s)$. This is the stationary distribution of the Markov chain formed by the feasible solutions s of our system (e.g., all possible L-node subsets in the network). In order to ensure that the Markov chain has a unique stationary distribution, the Markov chain must be ergodic. Usually, this distribution is chosen such that it maximizes a desired utility.
- *Proposal density*: $Q(s, s')$. This is the probability that governs the "creation" of new solutions s'. In Opportunistic Networks, this is the probability of two nodes meeting each other.

From these, a new solution is accepted with probability:

$$p(s, s') = \min\left(1, \frac{\pi(s')}{\pi(s)} \cdot \frac{Q(s', s)}{Q(s, s')}\right) \tag{5}$$

As mentioned, the *proposal density* is imposed by the meeting probabilities. In general, the probability of two nodes meeting in an Opportunistic Network is symmetric. Thus, the term $\frac{Q(s',s)}{Q(s,s')}$ in equation 5 cancels. Hence, it is not necessary to know the contact probabilities between nodes, as long as they are symmetric. This is very convenient, since one can usually only get approximations of the probabilities.

Thus, the acceptance probability of a new solution depends only on the distribution to be sampled, $\pi(s)$:

$$p(s, s') = \min\left(1, \frac{\pi(s')}{\pi(s)}\right) \tag{6}$$

We must choose $\pi(s)$ in such a way that, with high probability, the Markov chain "walks" towards a solution with the highest utility. The Gibbs distribution has this property [3]:

$$\pi(s) = \frac{\exp\left(\frac{U(s)}{T}\right)}{\sum_{v \in \mathcal{S}} \exp\left(\frac{U(v)}{T}\right)}, \tag{7}$$

where $\mathcal{S}$ is the space of all possible configurations and T is a system parameter, the "temperature". When T is small, the distribution is concentrated around the large values of $U(s)$ and thus, the algorithm will converge to a "good" solution with high probability. Seen over time, this equivalently means that good configurations are in place for the majority of time. Note that calculating the normalizing constant in the denominator in equation 7 could be a potential problem. Nevertheless, the advantage of MCMC algorithms is that this cancels out in equation 6 and nodes in contact can locally calculate the following quantity:

$$p(s, s') = \min\left(1, \frac{\exp\left(\frac{U(s')}{T}\right)}{\exp\left(\frac{U(s)}{T}\right)}\right) = \min\left(1, \exp\left(\frac{U(s') - U(s)}{T}\right)\right) \tag{8}$$

Summarizing, the algorithm executed by each node on contact with another node, in the case of MCMC-based optimization is shown in Algorithm 2.

Such a Metropolis-Hastings algorithm using the Gibbs distribution as the stationary distribution is also known as simulated annealing.

Algorithm 2: `MCMC` for Optimal Relay Selection

Data: n_a applies `MCMC` upon meeting n_b. $\mathcal{L}_a$, $\mathcal{L}_b$ the set of relays with n_a, and resp. with n_b.
Result: For each message carried by n_a, potentially a new configuration.

```
2.1  for m ∈ Messages(n_a) do
2.2      U_a ← U_M(m,N,L_a);
2.3      U_b ← U_M(m,N,L_b);
2.4      if U_b ⩾ U_a then
2.5          Messages(n_a) ← Messages(n_a) \ m;
2.6          Messages(n_b) ← Messages(n_b) ∪ m;
2.7      else
2.8          p ← exp((U_b − U_a)/T);
2.9          if p ⩾ rand(0,1) then
2.10             Messages(n_a) ← Messages(n_a) \ m;
2.11             Messages(n_b) ← Messages(n_b) ∪ m;
```

4.2.2 The Role of the Temperature

As aforementioned, the temperature parameter in equation 7 controls the kurtosis or peakedness of the Gibbs distribution we use as the stationary distribution of our Markov chain. There is a tradeoff involved in the choice of T:

- ***T* → 0.** For very small T, the distribution has very sharp peaks at large values of $U(s)$. In theory, this means that, after a high enough number of steps the Markov chain will almost surely converge to the maximum $U(s)$. However, this also means that it will be very difficult to escape local maxima, since they too have high probabilility compared to neighbouring configurations. Because of this, the convergence time will be significantly increased.
- ***T* high.** For larger values of T, the distribution is rather flat even for high values of $U(s)$. Therefore, the Markov chain will easily escape local maxima and converge faster to the global maximum. However, when it does reach the global maximum, it will equally easily escape from it.

To overcome these difficulties, an adaptive T is ideal. This type of algorithm is often referred to as *simulated annealing*. The way in which the temperature is adapted is called a "cooling schedule": start with a relatively high T, so that the Markov chain is directed towards the region of high utility configurations and gradually cool down the system, in order to contain the chain to that region.

Cooling schedules are specific to each problem, and related to the mixing times of the respective Markov Chains. However, some very popular cooling schedules are used in a broad range of contexts. Assuming we start with an initial temperature value T_0, some examples of schedules are [20]: (i) linear schedule: $T(t) = T_0 - \eta t$, (ii) exponential schedule: $T(t) = T_0 \cdot \alpha^t$, $(\alpha < 1)$, (iii) logarithmic schedule: $T(t) = \frac{c}{\log(t+d)}$. While in theory, the logarithmic cooling with appropriate parameters is *guaranteed* to find a global optimum [7], in practice, it is often more efficient to use the empirically best suited cooling. In this paper, we will use an empirical exponential cooling schedule.

5. PERFORMANCE EVALUATION

In this section, we will present a preliminary evaluation of distributed optimization algorithms (greedy and stochastic) using a real mobility trace and a synthetic mobility model. With optimal relay selection as our case study, we will study

the convergence properties of these two algorithms towards the optimal solution.

5.1 Contact Generators

We use the following two contact generators:

TVCM Contacts: The TVC model [10] creates community-based mobility scenarios of tunable complexity, and has been shown to reproduce a number of properties observed in real traces. We use it here to create a scenario that includes local maxima for the optimal relay selection problem, as described in Section 4.1.2. This scenario (Figure 1) has 104 nodes and 1 000 000 contacts. The four communities contain respectively 47 nodes, 19 nodes, 19 nodes and 9 nodes[3]. In addition, there are 10 bridging nodes.

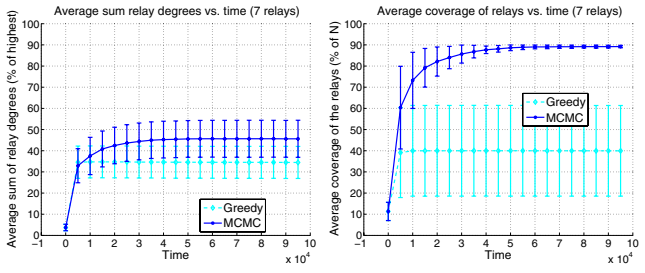
MIT Contacts: The second trace comes from the Reality Mining [6] project. 97 students and employees of MIT were equipped with mobile phones scanning every 5 minutes for Bluetooth devices in proximity during 9 months. This trace is unique in terms of number of devices and duration. Nevertheless, with a time granularity of 5 minutes, many short contacts were presumably not logged. For our simulations, we cut the trace at both ends and used 100 000 contacts reported between September 2004 and March 2005.

For both traces we ignored logged timing information and just ordered the reported contacts according to their start times (i.e., slotted contacts). We obtain the social graph from the contacts using the method discussed and the results (optimal parameters) obtained in [9].

5.2 Simulation Results

On these traces, we conducted experiments using both utilities U_1 and U_2. For each utility, we compared the performance of the two algorithms presented in Section 4. Due to space limitations, only a selection of these results is discussed in the following.

Figure 2 compares the convergence behavior of the two algorithms in the case of the TVC model, for 7 message replicas. We define *convergence* as the percentage of the optimum utility actually reached by each algorithm. Each plot represents the average over all messages from the creation of the message until its expiration (around 600 messages generated randomly throughout each trace; error bars measure the standard deviation).



(a) TVC model with U_1 (b) TVC model with U_2

Figure 2: Average convergence over the duration of a TTL in the TVC Model

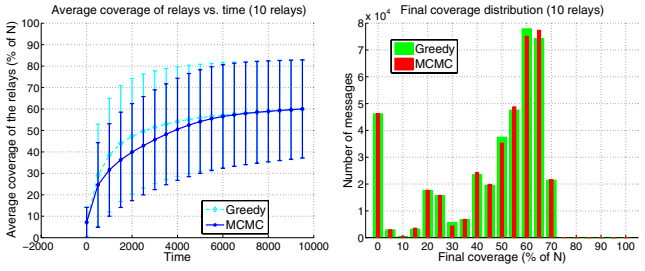
For both utilities the MCMC algorithm shows clear superiority over the greedy algorithm. As explained in Section 4.1.2, the TVCM scenario we are using has permanent local maxima by construction. Therefore, it is anticipated that the greedy algorithm will not converge to the maximum. The MCMC algorithm, on the other hand, is able to effectively navigate through the local maxima and reach considerably higher convergence[4]. Finally, we observe that the variance for the MCMC case is significantly smaller. This implies that the MCMC-based schemes have a better worst-case behavior, especially for larger numbers of replicas (this is expected due to the law of large numbers).

In Table 1, we further examine the effect of number of replicas (relays) used on the performance difference between the two algorithms for utility U_2. As shown there, when the number of replicas is smaller than the number of communities, neither algorithm can achieve 100% coverage, as expected. However, with 4 or more relays, MCMC converges quickly to near optimal coverage, significantly outperforming the greedy algorithm.

# relays	2	4	6	8	10
MCMC	45	78	87	90	90
Greedy	17	28	37	45	54

Table 1: Final convergence (% of N) in function of the number of carriers

We now turn our attention to the MIT trace. Figure 3 compares the two algorithms for utility U_2 (results for U_1 are similar and omitted due to space limitations). Figure 3(a) shows the average convergence calculated as above, for 10 relays. As the figure implies, *on average*, the two algorithms achieve comparable performance[5]. This is somewhat surprising, as it implies that, for most initial configurations (relays reached initially after spraying), there exist no (long-lasting) local maxima in the MIT trace. To get a better view, in Figure 3(b) we plot the empirical distribution for the utility of the reached configuration, by different messages. As is shown there, the two algorithms do appear to make some different decisions, with the MCMC algorithm managing to reach a better configuration for some messages. We noticed that is more prominent when the number of relays increases. This is reasonable, as this increases the chance that one of the replicas will get "stuck" in the greedy case.



(a) Average convergence (b) Convergence distribution

Figure 3: Convergence average and distribution for the MIT trace using U_2

[3]The choice of community sizes is random. Any values > 10 nodes per community produce qualitatively similar results.

[4]Yet, we have observed that not all MCMC-navigated messages reach an optimal configuration during the TTL. This is partly due to cooling that must be tuned to a scenario, or the need for longer time than the TTL to converge. In the future, we will look deeper into optimal cooling, in function of the mobility pattern.

[5]The y-axis now depicts the total number of nodes dominated by the reached relay set. Hence, a convergence of < 100% simply implies that perhaps more vertices are needed.

Summarizing, our preliminary implementation of a distributed optimization algorithm using MCMC methods seems to be able to converge to high quality configurations for the MIT trace as well, most of the time[6]. As mentioned earlier, performance could be further improved by carefully tuning the cooling scheme. At the same time, these results also suggest that simple greedy algorithms might suffice to achieve good performance in some real mobility scenarios, due to favorable *mixing properties*. However, we cannot yet generalize before considering additional traces, and larger networks (where some local maxima are more probable to appear).

Finally, we turn back to the TVCM scenario and compare the performance of the two utility functions *with respect to the achieved coverage rather than optimality*, in Figure 4.

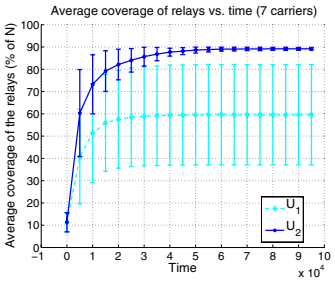


Figure 4: Comparison between coverages achieved by the MCMC algorithm with utilities U_1 and U_2

When using U_1 both algorithms will try to send each replica to high degree nodes (most of which reside in the largest community). While both algorithms do a decent job for *this task* (as shown in Figure 2(a)), when considering our content placement case study, sending all copies in the same community is clearly sub-optimal. U_2 attempts to correct this and can achieve a much better performance for the problem in hand, as is evident in this figure.

6. CONCLUSION

In this paper, we have looked into the problem of distributed optimization for Opportunistic Networks. We have argued that stochastic optimization algorithms, namely the framework of Markov Chain Monte Carlo (MCMC) optimization, are a natural fit for this type of problems. We have used the problem of optimal relay selection for group communication (e.g., multicast), as a case study, and have shown that MCMC-based algorithms are able to evade local maxima and converge to high utility configurations. At the same time, we have also observed that simple (deterministic) "gradient-ascent" type of algorithms often perform well, despite the non-trivial contact patterns observed in collected traces. We believe this is due to the dynamicity of local search neighborhoods, which while static and fixed for traditional optimization problems, are constantly changing here due to node mobility. In future work, we intend to analytically study both greedy and MCMC-based optimization algorithms, and derive appropriate predictors for their performance, as a function of the mobility pattern and utility function in hand.

[6]These traces are known to have skewed degree distributions, so random choice of relays cannot lead to the high coverage values observed here.

7. REFERENCES

[1] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN routing as a resource allocation problem. *SIGCOMM Comput. Commun. Rev.*, 37(4):373–384, 2007.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[3] P. Brémaud. *Markov chains : Gibbs fields, Monte Carlo simulation, and queues*. Springer, Berlin, Germany, 2001.

[4] M. Chen, S. C. Liew, Z. Shao, and C. Kai. Markov approximation for combinatorial network optimization. In *Infocom 2010*, pages 1–9.

[5] E. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant MANETs. In *MobiHoc 2007*.

[6] N. Eagle and A. Pentland. Reality mining: sensing complex social systems. *Pers. Ubiq. Comput.*, 10(4):255–268, 2006.

[7] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the bayesian restoration of images. *Journal of Applied Statistics*, 20(5):25–62, 1993.

[8] A. Guerrieri, A. Montresor, I. Carreras, F. D. Pellegrini, and D. Miorandi. Distributed estimation of global parameters in DTNs. In *IEEE WoWMoM AOC 2009*.

[9] T. Hossmann, T. Spyropoulos, and F. Legendre. Know Thy Neighbor: Towards Optimal Mapping of Contacts to Social Graphs for DTN Routing. In *Infocom 2010*, pages 1–9.

[10] W.-J. Hsu, T. Spyropoulos, K. Psounis, and A. Helmy. Modeling spatial and temporal dependencies of user mobility in wireless mobile networks. *IEEE/ACM Transactions on Networking*, 17(5):1564 –1577, Oct 2009.

[11] L. Hu, J.-Y. Le Boudec, and M. Vojnović. Optimal channel choice for collaborative ad-hoc dissemination. In *Infocom 2010*.

[12] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *MobiHoc 2008*.

[13] B. Johansson, M. Rabi, and M. Johansson. A simple peer-to-peer algorithm for distributed optimization in sensor networks. In *IEEE CDC 2007*, pages 4705–4710.

[14] B. Kauffmann, F. Baccelli, A. Chaintreau, V. Mhatre, K. Papagiannaki, and C. Diot. Measurement-based self organization of interfering 802.11 wireless access networks. In *Infocom 2007*, pages 1451–1459.

[15] J. Kneis, D. Mölle, and P. Rossmanith. Partial vs. Complete Domination: t-Dominating Set. In *SOFSEM 2007: Theory and Practice of Comp. Sc.*, pages 367–376.

[16] A. Krifa, C. Barakat, and T. Spyropoulos. Optimal buffer management policies for delay tolerant networks. In *IEEE SECON*, pages 260–268, 2008.

[17] V. Lenders, M. May, G. Karlsson, and C. Wacha. Wireless ad hoc podcasting. *SIGMOBILE Mob. Comput. Commun. Rev.*, 12(1):65–67, 2008.

[18] V. Lenders, J. Wagner, and M. May. Measurements from an 802.11b mobile ad hoc network. In *WOWMOM 2006*.

[19] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, 2003.

[20] Y. Nourani and B. Andresen. A comparison of simulated annealing cooling strategies. *Journal of Physics A: Mathematical and General*, 31(41):8373, 1998.

[21] A. Picu and T. Spyropoulos. Minimum Expected *-cast Time in DTNs. In *BIONETICS 2009*.

[22] O. Sandberg. Distributed Routing in Small-World Networks. In *SIAM ALENEX 2006*.

[23] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *SIGCOMM WDTN 2005*.

[24] T. Weise, H. Skubch, M. Zapf, and K. Geihs. Global optimization algorithms and their application to distributed systems. Technical report, University of Kassel, 2008.

[25] Y. Yu, Y. Zhou, and S. Du. A Markov Chain Monte Carlo approach to perform global optimized resource replication over ad hoc networks. In *ACM WiCOM 2009*.