Self-Adjusting Networks to Minimize Expected Path Length

Chen Avin*, Michael Borokhovich*, Bernhard Haeupler† and Zvi Lotker*

*Ben-Gurion University of the Negev
Emails: {avin, borokhom, zvilo}@cse.bgu.ac.il
†Massachusetts Institute of Technology
Email: haeupler@mit.edu

Abstract—Given a network infrastructure (e.g., a grid, data-center or on-chip-network) and a distribution on the source-destination requests, the expected path (route) length is an important measure for the performance, efficiency and power consumption of the network. In this work we initiate a study on self-adjusting networks: networks that use local-distributed mechanisms to adjust the position of the nodes in the network to any distribution of route requests. Finding the optimal placement of nodes is defined as the minimum expected path length (MEPL) problem. While, as we show, the MEPL problem is NP-hard to solve exactly in general we give efficient and simple approximation algorithms for interesting and practically relevant special cases of the problem. E.g., we consider grid networks in which the distribution of requests is a symmetric product distribution. In this setting we show that simple greedy switches, which locally minimize an objective function, achieve good approximation ratios. We are able to prove this result using the useful notions of expected rank of the distribution and the expected distance to the center of the graph.

I. INTRODUCTION

In the last decade we have witnessed two new related phenomena in distributed computing. The first is the emerge of huge data centers and warehouse-scale computers. The second phenomenon is the decentralization and parallelism of workload in single computers. In both cases the system is a network of computing primitives that share global computational goals. A common problem to both data centers and single computers is their increasing power consumption. Energy waste is maybe among the most major problems of the 21st century. For example, the total cost of power consumption of data centers in the USA alone is estimate by 50 billion dollar [1]. Moreover, the energy consumed by data centers is estimate to double every five years [2]. Therefore, energy efficient communication networks has become a major problem to be improved upon.

One characteristic that is shared by both data centers and modern multiprocessor computers is the execution of multiple processes that run in parallel. In many cases these process need to communicate with each other to work on a shared task. It is estimated that in data centers the energy consumed by routing is about 20%-30% of the total energy consumed [3]. Routing in network-on-chip (NoC) consumes even up to 50% of the total energy [4]. These numbers pose our community both challenge and opportunity. The opportunity is to gain significant energy savings for these systems. The challenge is to design and implement clever and simple algorithms that can improve routing efficiency.

Another common property of these systems is that they all operate in a fixed network infrastructure. This mean that we cannot change the structure of the network by rewiring links. Instead, we can move the locations of processes between different machines (or CPUs). In this paper we formulate the above guideline to study the problem of saving energy on a fixed infrastructure. The basic idea is that energy cost of routing in a network is proportional to the length of the routes. This suggests the following idea: If we can make the routes shorter then we save energy. In this work, we offer local and distributed algorithms that (re-)place processes in the network to reduce the expected path length between nodes and therefore reduce the communication energy cost.

The problem of minimizing the total energy consumed by routing is dependent on two major properties of the system: the structure of the communication network and the statistical pattern of route requests, i.e., sources and destinations. We first show that even in a very simple pattern such as every nodes having an activity level and the probability to send or receive a message is proportional to its level, the problem is NP-hard on general network topologies. Such routing and activity distributions are partially justified from real data [5], [6]. On the other end, when the network is “simpler” or regular, like a grid network, the problem can still be hard if the pattern is in some sense complex. Since the problem is NP-hard we turn to analyze approximation algorithms. We concentrate on local and distributed algorithm, namely, processes can be exchanged (i.e., relocated) only between nearby nodes without any centralize coordination.

A. Overview of our results

First, we formulate the discussed problem as the minimum expected path length (MEPL) problem, that is, given a network infrastructure and a distribution of requests, minimizing route costs by finding an optimal placement for processes (nodes). We then show that MEPL is NP-hard in general, either by the complexity of the network or the distribution of requests. Next we present a simple, local, distributed algorithms that achieves good approximation to the optimal solution for the MEPL problem for some specific topologies and requests distribution. Our algorithm is “self-adjustable” in the sense that nodes...
switch processes based on the continuously observed sequence of route requests each node involved in in a similar spirit of splay trees [7]. In particular we are able to show (informal):

**Theorem A.** For a d-dimensional grid network and a symmetric product distribution of requests there is a simple distributed algorithm, which defines a local switch policy between processes and its neighbors and achieves a constant approximation to the minimum expected path length (MEPL) problem.

Interestingly, we prove this theorem using a measure called expected rank which is related to the uncertainty of a random variable in a similar manner as entropy is.

We then turn to more complex distributions of requests and discuss requests that are clustered into disjoint groups. In such a setting our local algorithms are not guaranteed to always perform well anymore. Nevertheless we present promising simulation results that consider random initial states of processes locations in the network. For the 2-dimensional grid, our local algorithms seems to perform very well.

**Organization:** In Section II we discuss related work and somewhat similar approaches. Section III introduces the formal problem and definitions. The hardness of MEPL is proved in Section IV and then in Section V we prove constant approximation in grids with product route distributions. We end the paper with a discussion and simulations on the clustered requests setting in Section VI.

II. RELATED WORK

Energy saving along with green computing is an active topic of research in the recent years. In a recent paper [8] Lis et al. study memory architectures of microprocessor. The authors suggest that processes will migrate to a location that is closer to the data instead of what is common in today architectures, i.e., coping the data to be closer to the process. The logic behind this idea is that programs are much smaller than their data. We take this idea one step further by reducing the communication distance between two processes. Improving the energy efficiency of routing in networks was also considered. Batista et al. [9] used traffic engineering on grids to self-adjust to routing requests. In [3] and [10] different authors considered data centers and tried to save energy by powers down routers and links when demand in the network is low. Other self-adjusting routing scheme were considered, for example in scale-free networks to overcome congestion [11], [12].

To the best of our knowledge, there is no previous work on the minimum expected path length problem. In this respect, the most similar (and influential) to our work is the seminal work of Sleator and Tarjan on splay trees [7]. If one look on the tree as a network, splay trees self-adjust so to minimize the expected path length in the network if the request distribution is such where the source is a single (virtual) node that is connected to the root. In our future work we would like to consider the MEPL problem on binary search trees with additional types request distributions. Huffman tree [13] is also a solution to the MEPL problem if the distribution is known and routes only from the root to the leaves of the tree.

The local greedy switch strategy we use is related to physics and natural dynamics which indirectly try to minimize energy. Using this analogy for optimization purposes has a long history. E.g., simulated annealing [14] can be seen as simulating physics while cooling the temperature, i.e., the local moves selected shift over time more and more bias from mostly random behavior to greedy energy minimization. Here we only look at greedy steps. In networking context similar approaches were used for example for load balancing via diffusive paradigms [15] and for routing via gradient mechanism [16].

Another very related research is about self-stabilizing graphs [17], [18]. The goal there is also to maintain some objective using local edge exchanges, mostly in an overlay network. In a similar manner we would like to extend the current work to solve MEPL on overlay and peer-to-peer networks, using edge rewiring as well.

III. MODEL AND PROBLEM DEFINITION

We model the communication network by an undirected, unweighted and connected graph G = (V, E), where V is the node set and E is the edge set. We denote the number of nodes with n = |V| and the number of edges with m = |E|. We denote with d(·) the distance function between nodes in G, i.e., for two nodes v, u ∈ V we define dG(u, v) to be the number of edges of a shortest path between u and v in G.

We assume the network serves route requests drawn independently from an arbitrary distribution and messages are routed along a shortest paths. Let M ∈ V × V be a random variable that denote a route request for a message from node u to v. Let P be a probability distribution on M, namely on the routes requests and for every u, v ∈ V, let p(u, v) denote the probability of the request {u, v}. We can now define the expected path length:

**Definition 1 (expected path length).** Given a network G and a probability distribution P the expected path length of G and P is defined as:

\[
EPL = \mathbb{E}_{G, P}[d(M)] = \sum_{u, v \in V} p(u, v) \cdot d_G(u, v) \quad (1)
\]

We drop the subscripts G and/or P when they are clear from the context. Note that a special case of this definition is when P is the uniform distribution. This gives the average path length which is often used in the literature instead of the diameter of the network, for example to show that a network is a small world network [19].

Given a network infrastructure G and a distribution on the request we would like to find optimal placing of the nodes in the network so to minimize the expected path length. A **placement** (or **switching**) function is a a one-to-one and onto function ϕ(V), ϕ : V → V. This function determine the new locations of nodes in the network. We write ϕ(G) to denote a new graph G′ = (V, E′) where u, v is an edge in G if and only
if \( \varphi(u), \varphi(v) \) is an edge in \( G' \). Given a placement function \( \varphi \), \( G \) and \( \mathcal{P} \) the expected path length:

\[
EPL(\varphi) = \sum_{u,v \in V} p(u,v) \cdot d_{\varphi(G)}(u,v)
\]

Note that the activity level of nodes names remain the same only there location at the infrastructure (i.e., \( G \)) change. For \( G \) and \( \mathcal{P} \) we would like to find the placement that gives the minimum expected path length (MEPL). Formally:

**Definition 2 (Minimum Expected Path Length problem).** Given a graph \( G \) and a probability distribution \( \mathcal{P} \) on \( V \times V \) find a placement function, that minimize the expected path length:

\[
MEPL = \min_{\varphi} EPL(\varphi) \tag{2}
\]

As mentioned earlier this problem is motivated by the network serving point-to-point routing requests that are independently sampled from a distribution \( \mathcal{P} \). If we assume that the cost for a request \( u,v \) is \( d(u,v) \) then the MEPL problem tries to minimize the expected cost of a route. Note that this is also equivalent to minimizing the expected usage of all links or minimizing the expected total number of transmissions - all important metrics in terms of energy saving and efficiency.

In this work we mostly consider local and distributed switching rules to find a good placement: rules where a node is only allowed to switch places with nodes that are in its neighborhood (i.e., closed to it). The goal is that after a sequence of local switches the network will reach its minimum expected path length and solve the MEPL problem.

As it appear, this is not always practical, in the next section we prove that in general settings (of \( G \) and \( \mathcal{P} \)) and even with global knowledge and switching, solving the MEPL problem is hard.

Throughout the paper we consider at times simpler forms of networks and requests distributions, i.e., grid networks and the symmetric product distribution:

**Definition 3 (d-dimensional grid networks).** A mesh network of size \( n = k^d \) with nodes embedded on all locations \( [k]^d \) where \( [k] \) is the set of integers \( 1, 2, \ldots, k \). Each node is connected to all the nodes at one \( \ell_1 \) distance from it, i.e., at most 2 in each of the \( d \) dimensions.

**Definition 4 (symmetric product distribution).** In symmetric product distribution, each node has a level of activity and the more two nodes \( u \) and \( v \) are active the more likely that the route \( \{u,v\} \) gets requested. Moreover, we scale the activity levels of the nodes such that they form a distribution with an activity level \( p(u) \) for each node \( u \) and assume that the request distribution is induced by the cross product of the activity levels, i.e., \( p(u,v) = p(u) \cdot p(v) \).

Next we show that even with symmetric product distribution MEPL is hard.

## IV. Hardness of MEPL

In this section we prove two results that demonstrate how the hardmess of the problem can come form either the graph \( G \) or the structure of the routing request distribution \( \mathcal{P} \). This serves also as an additional motivation why in the rest of the paper we turn to graphs and distributions with more (realistic) structure.

For both our examples it suffices to use probability distributions that only have one non-zero probability value. In our first statement we even show that further restricting oneself to symmetric product distributions the MEPL problem is hard on general networks:

**Lemma 1.** Given a graph \( G \) and a symmetric product distributions \( \mathcal{P} \) it is NP-hard to decide whether the MEPL is larger or smaller than a given value.

**Proof:** We reduce this problem to the well known \( k \)-CLIQUE problem. For this we use a symmetric product probability distribution that puts \( k^{-2} \) probability weight on each of the pairs \( V' \times V' \) formed by a subset of the nodes of size \( k \) and zero probability on any other pair. The unique optimal solution to MEPL with value \( 1 - 1/k \) will be obtained if all \( k \) nodes are placed in a subgraph of \( G \) that induces a \( k \)-CLIQUE. Deciding whether a graph \( G \) contains such a subgraph and thus also deciding whether a solution with value \( 1 - 1/k \) is achievable is therefore NP-hard [20].

Even if we restrict the graph to be nice, e.g., the 2-dimensional grid graphs a lack of structure in the probability distribution can make the MELP problem hard:

**Lemma 2.** Given a probability distribution \( \mathcal{P} \) it is NP-hard to decide whether the MEPL on a 2-dimensional grid is larger or smaller than a given value.

**Proof:** We reduce this problem to the problem of embedding a tree in a 2-dimensional grid which was shown to be NP-hard by Bhatt and Cosmadakis [21]. More precisely it is NP-hard to decide whether a given tree (with maximum degree 4) is a subgraph of the grid. To construct a family of hard probability distributions we simply create for every tree on \( n \) nodes a distribution that puts a probability mass of \( 1/(n-1) \) on each pair of nodes that are neighbors in the tree and zero on all other pairs. The MELP for such a distribution in the \( n \) times \( n \) grid is 1 if and only if the tree can be embedded in the grid. If this is not the case at least one path will be of length two raising the average above 1.

In the next sections we will show that if one assumes a grid graph and a symmetric product distribution nice algorithmic results can be obtained.

## V. MEPL with Symmetric Product Distributions

For general request distribution it is hard to find a good or optimal solution even when one is not restricted to local and distributed switching rules. So, first, we restrict ourselves to a simpler model of requests – symmetric product distribution (Definition 4). Second, we assume \( d \)-dimensional grid topologies, and in particular the line and a 2-dimensional grid.
We assume that a node knows the distribution of requests it involved in and thus it can decide whether the switching (exchanging positions) with a neighbor will increase or decrease the objective function, the expected path length of the network.

The main result of this section is that under the above settings a good approximation to the objective function can be found using only a simple (greedy) local switching rules!

To prove these results we need the following important definitions.

A. Expected Distance to Center and Expected Rank

To find a good placement for nodes which gives a good approximation to the MEPL we define the expected center and the expected distance to it.

**Definition 5 (center and expected distance to the center).** The expected center of a graph $G$ and a symmetric product distribution $\mathcal{P}$ is a node $c$ s.t.:

$$ c^* = \arg \min_{x \in V} \sum_{u \in V} p(u) d(u, x) $$

The expected distance to the center $c^*$ is:

$$ C = \min_{x \in V} \sum_{u \in V} p(u) d(u, x) $$

or equally

$$ C = \sum_{u \in V} p(u) d(u, c^*) $$

Both $C$ and $c^*$ can be written as functions of a placement $\varphi$, i.e., $C(\varphi)$ and $c^*(\varphi)$, when $G$ and $\mathcal{P}$ are clear. The minimum expected distance to the center is defined as $C_{\min}$:

$$ C_{\min} = \min_{\varphi} C(\varphi) $$

The next lemma describe the relation between $C$ and MEPL.

**Lemma 3.** For any given placement $\varphi$,

$$ 2C(\varphi) \geq \text{EPL}(\varphi) \geq C(\varphi) $$

**Proof:** To see the upper bound we suppose that instead of routing between two nodes directly we route every request via the location of $c^*$. Routing a request in this way results in sampling two requests and summing up their distances to the center. In expectation this is exactly $2C$. Formally (for any $\varphi$):

$$ \text{EPL} = \sum_{u, v \in V} p(u)p(v)d(u, v) $$

$$ \leq \sum_{u, v \in V} p(u)p(v)(d(u, c^*) + d(c^*, v)) $$

$$ = \sum_{u \in V} p(u)d(u, c^*) + \sum_{v \in V} p(v)d(c^*, v) = 2C $$

The fact that the $C$ is a lower bound we show as follows:

$$ \text{EPL} = \sum_{u, v \in V} p(u)p(v)d(u, v) = \sum_{u \in V} p(u) \sum_{v \in V} p(v)d(u, v) $$

$$ \geq \sum_{v \in V} p(v) \sum_{u \in V} p(u)d(c^*, v) = \sum_{v \in V} p(v)d(c^*, v) = C $$

An important ingredient in bounding the performance of our local rules will be the following measure of expected rank. This quantity is an interesting measure on the concentration and uncertainty of a distribution.

**Definition 6 (Rank of nodes and the Expected rank).** The rank of a node is the position of the node in the ordered list of nodes’ probabilities (breaking ties arbitrarily). The node with the highest probability has rank $0$. The rank of the node $u \in V$ is denoted as $r(u)$. The expected rank of a probability distribution on the nodes of graph $G$ is

$$ \mathbb{E}[R] = \sum_{u \in V} p(u)r(u) $$

We next describe the local switching rules by which our distributed algorithm works.

B. (Greedy) Local Switching Strategies

We propose the following greedy strategy. A node switches with a neighbor if according to the (observed) marginal distribution on the requests involving itself and its neighbor switching positions improves the objective value. In this work we consider two simple optimization rules:

1) M-rule: Node will switch locations with its neighbor if the switch will minimize the objective function: the expected path-length between all pairs of nodes. This criterion is exactly the MEPL objective.

2) C-rule: Node will switch location with its neighbors if the switch will minimize the expected path-length between the center node and all the other nodes. This objective does not give us a solution for the MEPL problem, but it will be used as an upper bound for it.

If nodes switch only if this decreases the expected path-length (or some other criterion) then it is clear that this strictly monotone potential can not drop indefinitely (or too often) and thus a (quick) convergence is guaranteed. A placement $\varphi$ is said to be local minimum (or local optimum) iff no node in $G$ can switch according to the rule they operate (i.e., M-rule or C-rule). When using the C-rule we can prove the following about local minimum placement.

**Lemma 4.** Any local minimum in C-rule is center monotone: If $u$ is a neighbor of $v$ on a shortest path from $v$ to $c^*$ than $p(u) \geq p(v)$.

**Proof:** Assume by contradiction that $\varphi$ is a local minimum and $u$ is a neighbor of $v$ on a shortest path from $v$ to $c^*$ but $p(u) < p(v)$. Let $\varphi'$ be the placement after the switch, we will show $C(\varphi') < C(\varphi)$ contradiction to the local optimality of $\varphi$. Let $C'(\varphi')$ be the expected distance to $c^*(\varphi)$ (old center) in $\varphi'$ (new placement). Clearly $C'(\varphi') < C(\varphi)$ since we changed only two routes (of $u$ and $v$) and $v$ which has higher probability got closer. But now, by definition $C(\varphi') \leq C'(\varphi')$ and we done.

Note that according to the C-rule two neighbors switch location only if the switch decreases $C$: the expected distance
to the center. The improvement of the switch can be found locally, since the center location can be computed locally at each node via the expected position of its requests (which are identical to all nodes because of the product distribution). Therefore the C-rule will greedily minimize for each node the distance to the expected position of its requests.

Throughout the rest of this paper we assume that the system converges against a local minimum and analyze the performance of such a solution in this stable state. On the other hand we do NOT assume anything about the starting position or the specific order of the dynamics (node switches). Thus, in many cases an initially random starting position converges (e.g., using random improving switches) to a (near) optimal solution, we make no such assumptions and assume a worst case sequence of improvements and a worst-case initialization.

C. The Line - Linear placement

First, we study a greedy local switch strategy on a 1-dimensional grid - the line. We assume that the C-rule switching strategy is sequentially applied (in arbitrary order) on an arbitrary initial state and continuously adjust the network by switching neighbors. The strategy will converge against a local optimum from which no switch of two neighboring nodes improves the objective value in expectation. We are interested in quantifying how far such a locally optimal solution can be from the global optimum. The following theorem gives an answer for this question.

**Theorem 1.** Let $G$ be the line and $P$ a symmetric product distribution, then any locally optimal solution achieved by the C-rule (or M-rule) is at most a factor of 4 larger than the global optimum of MEPL.

We prove this theorem for the C-rule but this could be done similarly to the M-rule. Assume $G$ and $P$ as in the theorem. We first give an upper bound on the expected path length achieved by the C-rule in terms of the expected rank of the distribution (Definition 5).

**Lemma 5.** For any locally optimal solution $\varphi$ achieved by the C-rule on $C(\varphi) < E[R]$ and $EPL(\varphi)$ is at most $2E[R]$.

**Proof:** $d(v, c^*)$ is the distance of $v$ from $c^*$ in $\varphi$ and we what to bound it in terms of $r_v$, the rank of $v$.

From Lemma 4 all nodes between $v$ and $c^*$ on the line have higher probability than $v$ so $d(v, c^*) \leq r_v$.

$$C = \sum_{v \in V} p(v)d(v, c^*) \leq \sum_{v \in V} p(v)r(v) = E[R]$$

From Lemma 3 $EPL(\varphi) \leq 2C(\varphi) \leq 2E[R]$. We now prove a lower bound for MEPL on the line and any symmetric product distribution of requests.

**Lemma 6.** $MEPL \geq C_{min} \geq \frac{1}{2}E[R]$.

**Proof:** Let $\varphi^*$ be the placement such that $C_{min} = C(\varphi^*)$. Note that by definition $\varphi^*$ minimize the expected path to the center. Given the center $c^*(\varphi^*)$ and an arbitrary node $v$ with a distance $d(v, c^*)$ from $c^*$, we want to find an upper bound on the rank of $v$ by showing how many nodes can have a larger activity level than $v$. Clearly, all such nodes will be at most at the distance $d(v, c^*)$ from the center, since otherwise $\varphi^*$ will not be optimal and $v$ will be switch to a worse location. Since there at most 2 nodes at distance $i$ from the center $d(v, c^*) \geq r(v)/2$, and thus:

$$C_{min} = \sum_{v \in V} p(v)d(v, c^*) \geq \sum_{v} p(v)r(v)/2 = \mathbb{E}[R/2]$$

To conclude the proof of Theorem 1 we combine Lemmas 3, 5 and 6 to get that for a local minimum $\varphi$:

$$2\mathbb{E}[R] \geq EPL(\varphi) \geq MEPL \geq \frac{1}{2} \mathbb{E}[R]$$

Thus, the ratio between the worst case local solution and the optimal solution is at most 4.

D. The $d$-Dimensional Grid

In this section we extend the ideas form the line to a grid network. Our results extend easily to grids of arbitrary dimension but for sake of simplicity we stick with the two dimensional case in which each internal node is connected to the four neighbors of $\ell_1$-distance 1.

We first show that using the same greedy approach as in the line, namely switching neighboring nodes using the M-rule, leads to a significantly worse ratio between local and global minimum.

**Lemma 7.** There is a local minimum with regards to the M-rule that is a factor of $\Omega(n^{1/4})$ worse than the global minimum.

**Proof Sketch:** Take $\sqrt{n}$ nodes with uniform probability $p = 1/\sqrt{n}$ and set all other nodes to probability zero. Arranging the nodes on a line is a local minimum since there are no improvement switches, and the expected path length is $\sqrt{n}/2$. For a close to optimal solution we arrange the nodes on a ball of radius $O(n^{1/4})$ around one node. The expected path length within the ball is also within a constant factor of the radius, i.e., $O(n^{1/4})$. This proves that the worst-case to best-case ratio is at least $\Omega(n^{1/4})$.

Surprisingly, we can avoid this locally stable but highly suboptimal solution by allowing slightly longer switches. The rule we propose is that a node can also switch with any of the neighbors in $\ell_1$-distance 3 that differs 2 in one axis and 1 in the other (similar to the chess knight moves) and the switching is according to the C-rule. In this case we can prove the following.

**Theorem 2.** Let $G$ be the 2-dimensional grid and $P$ a symmetric product distribution, then any locally optimal solution achieved by the C-rule (and allowing knight move switches) is at most a factor of 4.62 larger than the global optimum of MEPL.

The proof of this theorem is similar in spirit to the 1-dimensional grid, we provide bound on $C$ for the optimal and locally optimal placement.
ratio 2 : 1 and the longest leg equals $m$, can be calculated by the following formula:

$$A(m) := \sum_{i=0}^{\lfloor m/2 \rfloor} (\lfloor m \rfloor + 1 - 2i) \geq \frac{m^2}{4} + 1$$

Now we are ready to calculate the number of nodes inside the polygon $- S_{total}$. We denote the number of nodes in the middle rectangle as $S_1$, the number of nodes in the upper triangle as $S_2$ and so on according to Figure V-D.

So, we obtain that $S_{total} = S_1 + S_2 + S_3 + S_4 + S_5 - 3$, where $-3$ is needed since the node $v$ should not be calculated (but was counted twice) and $c^*$ should be calculated once (but was counted twice). By adding up the expressions we obtain:

$$S_{total} = S_1 + S_2 + S_3 + S_4 + S_5 - 3$$

Let’s look at the node $x_1$ (located 1 north and 2 west from $v$). Clearly, $x_1$ is closer to $c^*$ than $v$, and thus, according to Lemma 4, $p(x_1) \geq p(v)$. For the nodes that are close to the boundary of the polygon holds:

$$d(v, c^*) > d(x_1, c^*) > d(x_2, c^*) > \ldots > d(x_7, c^*)$$

and

$$d(v, c^*) > d(z_1, c^*) > d(z_2, c^*) > \ldots > d(x_7, c^*)$$

Clearly, the distances of other nodes inside the polygon to the center are also smaller that $d(v, c^*)$.

We are interesting in finding the number of nodes that have larger probability than $v$. So, what we have to do is to calculate the number of nodes that are bounded by the polygon. The number of nodes bounded by a right triangle shape with a legs

\[ \text{Lemma 8. For any local minimum of the C-rule } C(\varphi) \leq \frac{1}{\sqrt{6}}E[\sqrt{R}] \]
8 nodes at the distance 2, $4 \cdot d(v, c^*)$ nodes at the distance $d(v, c^*)$. So, we can write:

$$r(v) \leq \sum_{i=1}^{d(v, c^*)} 4i = 2(d(v, c^*))^2$$

So, we obtain $d(v, c^*) \geq \sqrt{r(v)/2}$, and thus:

$$C = \sum_{v \in V} p(v)d(v, c^*) = \sum_{v \in V} p(v)d(v, c^*)$$

$$\geq \sum_{v} p(v)\sqrt{r(v)/2} = \mathbb{E} [\sqrt{R/2}]$$

Now we are ready to prove the result of Theorem 2. From Lemma 3 and 9 we get that the minimum expected path length is at least $\mathbb{E} [\sqrt{R/2}]$. From Lemma 8 we obtain that $C(\varphi) \leq \mathbb{E} [\frac{4}{\sqrt{m}} \sqrt{R}]$, and thus by Lemma 3 the expected path length of a local minimum can not be larger than $2\mathbb{E} [\frac{1}{\sqrt{m}} \sqrt{R}]$. Therefore the ratio between the optimal solution and any achievable approximate solution (using greedy C-rule strategy) is: $2 \cdot \sqrt{\frac{m}{2}} \approx 4.62$.

### VI. Clustered Requests

To go away from the product distributions which essentially makes nodes uniformly compete for the same spots we look at a situation where our dynamics achieve a different goal then just sorting nodes with different priorities around the common center of attraction.

One such interesting situation is when the requests form different interest groups that predominantly communicate within a group. Such a locality of requests is quite common. What we would expect from a good solution is that this grouping is detected and the nodes belonging to a group get a separate part of the network in which the requests between them get routed quickly without going over nodes that are never interested in this. Such a well clustered assignment of groups to parts of the network can be a drastic improvement over a random initialization.

#### A. Clustered Route Requests: Definition

To formally model a clustered request distribution we assume for simplicity that it is a mixture of product distributions on disjoint supports. In the case that the product distributions are uniform this results in the following. Let $k$ be the number of groups and let $V_1, \ldots, V_k$ be the vertex disjoint node sets of these groups and let $V_0 = V \setminus \bigcup_i V_i$ be the inactive nodes. Let $p_i$ for $1 \leq i \leq k$ be the activity level of each group such that $\sum_{i=1}^{k} p_i = 1$. Given this we have that the probability $p(u,v)$ for a request between a nodes $u,v \in V$ is:

$$p(u,v) = \begin{cases} p_i/|V_i|^2 & u,v \in V_i, i > 0 \\ 0 & \text{otherwise} \end{cases}$$

### B. Bad Topologies and Local Minima for Clustered Requests

We conjecture that on almost any topology there exists quite bad local minima for the clustered requests. The reason is that the cluster centers could coincide which makes all nodes want to go to the same direction. Thus a very mixed initial solution can be stable and will have long average paths for all clusters.

For example, on ring and torus we expect the center of $k$ randomly placed nodes in the plane to be very close to the middle. Thus, even initializing many clusters randomly on such a topology will most likely result in a bad local minimum.

In Section V we proved that under the assumption of symmetric product distribution of requests, in line and grid networks, the ratio between any local minimum solution and the optimal is at most 4. In this section we show that when we have clustered requests in a ring network, the ratio between the worst local minima and the optimal solution is at least $1.5c$, where $c$ is the number of clusters.

**Lemma 10.** In a ring network with uniform clustered distribution of requests, the ratio between the worst case nodes placement and the best case placement is at least $1.5c$, where $c$ is the number of clusters.

**Proof:** First we show that there exists a bad local minimum on the ring graph and uniform clustered distribution of request. We assume that there are $c$ clusters of $2k + 1$ nodes each, and thus the total number of nodes in a ring is $n = c(2k+1)$. Let $\varphi_i$ be the following placement of the nodes: $(v_1^1, v_2^1, v_1^2, \ldots, v_2^1, v_2^2, v_3^1, \ldots)$, where $v_i^j \in V_j$. We now prove that the placement of nodes $\varphi_i$ is locally optimal, i.e., there are no improvement switches.

Let’s denote the expected cost of the paths where the node $v$ is either source or destination as $E_v$. Due to the symmetric location and the uniform distribution, $E_v$ is the same for all the nodes $v \in V$. So, the expected path cost equals to $E_v$ for any $v \in V$. Let’s assume that $v \in V_1$, where $V_1$ is the set of nodes in cluster 1, and $|V_1| = 2k+1$. Let’s denote the probability of choosing the node $u \in V_1$ as a pair for the given node $v \in V_1$ as $p_{u|v}$. Clearly, $p_{u|v} = \frac{1}{|V_1|}$. Before the switch:

$$E_v = \sum_{u \in V_1} p_{u|v} d_{vu} = \frac{1}{|V_1|} \cdot 2 \sum_{i=1}^{k} c \cdot i = \frac{c \cdot k(k+1)}{2k+1}$$

Due to the symmetry, there is no difference with what node we will try to switch the node $v$. So, let’s assume that we switch it with its right-hand neighbor $-w$. In order to check the difference with the path cost before the switch, we need to consider all the paths in which nodes $v$ and $w$ are involved. Clearly, even after the switch $E'_v = E_v$ due to the symmetry, so we will now find the $E_v$ after the switch and will compare it to the $E_v$ before the switch. After the switch:

$$E'_v = \sum_{u \in V_1} p_{u|v} d_{vu}$$

$$= \frac{1}{|V_1|} \left( \sum_{i=1}^{k} (c \cdot i - 1) + \sum_{i=1}^{k} (c \cdot i + 1) \right) = \frac{c \cdot k(k+1)}{2k+1}$$
Since all possible switches are the same due to the symmetry, we can conclude that there is no switch that will improve the expected paths cost and the placement \( \phi \) is a local minima.

Now let’s assume different placement of the nodes \( \phi_{opt} = (v_1^1, v_2^1, \ldots, v_{|V_j|}^1, v_1^2, v_2^2, \ldots, v_{|V_j|}^2, \ldots) \), where \( v_i^j \in V_j \). For this, probably optimal, placement, we will calculate now the expected path length (EPL).

Let’s consider the part of the ring of \( 2k + 1 \) nodes that belong to a single cluster. Number of paths of length 1 inside the cluster is \( 2 \cdot 2k \). Number of paths of length 2 is \( 2 \cdot (2k - 1), \) length 3 is \( 2 \cdot (2k - 2), \) length \( m \) is \( 2 \cdot (2k - m + 1) \), and of length \( 2k \) is 2. So, we obtain the average path length in one cluster:

\[
E = \sum_{v, u \in V_i} p_{vu}d_{vu} = \frac{1}{2|V_i|^2} \cdot 2 \sum_{i=1}^{2k} m(2k - m + 1) \\
= \frac{k(2k + 1)(4k + 4)}{3 \cdot 2(2k + 1)^2} = \frac{k(2k + 2)}{3(2k + 1)}.
\]

C. Simulation Results

In this section we simulation results in 1 and 2-dimensional grids. In all simulations we assume a uniform clustered distribution of requests (Eq. (4)) but with uniform \( p_i \)’s.

The first simulation presented in Figure VI-C shows the improvement ratio between the EPL of a random starting locations to the EPL of a local minimum reached by the local greedy M-rule as a function of number of clusters. In the ring graph the improvement is almost negligible and it is constant with the number of clusters. Interestingly, when a greedy switching strategy is applied to a 2-dimensional torus graph, the improvement ratio is much better and it is growing with the number of cluster. This is a result on the higher connectivity of the 2-dimensional grid vs. the ring. This lead to the conjecture that this approach might work well in highly connected networks. In the grid the improvement also depend on the size of the network and for many small clusters it should grow as \( O(\sqrt{n}) \).

The next figure shows a specific example for the improvement in 2-dimensional torus. Overall the greedy switching algorithm successfully detects clusters and groups them together pretty well. However, there are still some clusters that are not connected. In Figure 3 (a) we can see the initial random placement of the clustered nodes. Nodes surrounded by a black rectangle are the centers of their clusters (as was defined earlier, a center is a node that has the minimal expected distance to all other nodes in the cluster). In Figure 3 (b) we see the placement of the nodes achieved by the greedy M-rule switching strategy. Although it looks that the nodes are highly grouped, we can see that the shapes of the clusters are not optimal (an optimal placement should look like a circle around the center of the cluster). Some clusters are stretched (e.g., brown cluster) and some are even not connected (e.g., orange cluster).

When every node on the torus belongs to an active cluster, we can frequently run into situation in which two nodes will not switch even if it is improvement for one of the clusters. This suboptimal local solution can be improved if we allow some nodes on a torus to be inactive. In the following two figures we see the results of such simulation where 50% on the nodes are inactive. In Figure 3 (c) we can see the initial random placement of the clustered nodes, (the inactive nodes are white colored). In Figure 3 (d) we see the placement of a local minimum of EPL achieved by the same greedy switching strategy. But now we can observe much nicer concentration of the nodes around their centers. These figures lead to many interesting future research questions about the topic. At the end of the paper there is a QR link to the animated version of the figures.

\[ \text{REFERENCES} \]


Fig. 3. (a) Torus with 900 nodes, 16 clusters. Uniform requests distribution. Initial state – random locations. There is no nodes in the inactive cluster \( V_0 \). (b) Final placement after applying local greedy strategy. All clusters are grouped together, but their shapes are not optimal for a given cluster. (c) Torus with 900 nodes, 8 active clusters. Inactive cluster \( V_0 \) has \( n/2 \) nodes. Uniform requests distribution. Initial state – random locations. (d) Final placement after applying local greedy strategy. All clusters are highly grouped together. All clusters are highly grouped together and their shapes are much closer to optimal. (e) Link to the animation of this simulation (http://www.bgu.ac.il/~avin/pmwiki/pmwiki.php?n=Main.Self-AdjustingNetworks).


