Theoretical Computer Science 584 (2015) 91–102

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Self-adjusting grid networks to minimize expected path length

Chen Avin [a,*], Michael Borokhovich [a,*], Bernhard Haeupler [b,*], Zvi Lotker [a,*]

[a] *Ben-Gurion University of the Negev, Israel*
[b] *Massachusetts Institute of Technology, USA*

A R T I C L E   I N F O

Article history:
Received 17 November 2013
Received in revised form 28 October 2014
Accepted 29 November 2014
Available online 5 December 2014

Keywords:
Self-adjusting networks
Distributed algorithms
Routing
Approximation
Minimum-linear-arrangement
Energy-efficient

A B S T R A C T

Given a network infrastructure (e.g., 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 (e.g., virtual machines) in the network to best fit the route requests distribution. Finding the optimal placement of nodes is defined as the minimum expected path length (MEPL) problem. This is a generalization of the minimum linear arrangement (MLA) problem where the network infrastructure is a line and the computation is done centrally. In contrast to previous work, we study the distributed version and give efficient and simple approximation algorithms for interesting and practically relevant special cases of the problem. In particular, we consider grid networks in which the distribution of requests is a symmetric product distribution. In this setting, we show that a simple greedy policy of position switching between neighboring nodes to locally minimize an objective function achieves 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.

1. Introduction

In the last decade we have witnessed two new major and related phenomena in distributed computing. The first is the emergence of huge data centers and warehouse-scale computers. The second phenomenon is the decentralization and parallelism of workload in single multi-core computers. In both cases (but on different scale) the system is a network of computing primitives that share global computational goals. In data center networks, as well as in modern multiprocessor computers, multiple processes run in parallel to execute common tasks so that, in many cases, these processes need to communicate with each other to work on their shared tasks.

Reducing energy waste, and in particular the power consumption of computing, is one of the major challenges of the 21st century. Data centers are no exception, and so is constantly increasing their energy and power usage. For example, the total cost of power consumption of data centers in the USA alone is estimated to be 50 billion dollars [1]. Moreover, the energy consumed by data centers is estimated to double every five years [2]. The focus of this work is to improve upon the energy that is consumed by routing in such systems. It is estimated that in data centers the energy consumed by routing is

* Corresponding authors.
E-mail addresses: avin@cse.bgu.ac.il (C. Avin), borokhom@cse.bgu.ac.il (M. Borokhovich), haeupler@mit.edu (B. Haeupler), zvilo@cse.bgu.ac.il (Z. Lotker).

http://dx.doi.org/10.1016/j.tcs.2014.11.036
0304-3975/© 2014 Elsevier B.V. All rights reserved.

about 20%–30% of the total energy [3]. Routing in a network-on-chip (NoC) consumes even up to 50% of the total energy [4]. These numbers pose to our community both an opportunity and a challenge. 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 means that we cannot change the structure of the network by, for example, rewiring links. But instead, what we can do, is to move the locations of processes (e.g., virtual machines) between the different computers (or CPUs). In this paper, we formulate the problem of saving energy on a fixed infrastructure network using migration of processes. The basic idea is that the energy cost of routing in a network is proportional to the length of the routes which suggests the following: If we can make the route lengths (or the *expected* route length) shorter, then we can save energy. We devise local and distributed algorithms that (re-)place processes in the network to reduce the expected path length. This can be achieved, for example, by Software Defined Networking (SDN) [5] – the concept, which provides, among others, much better control over the network functionality. In SDN, a software management platform may support an abstraction for moving a selected process from one physical machine to another. Recently, this approach became practical, when Google announced [6] the implementation of OpenFlow [7] in its own backbone.

The problem of minimizing the total energy consumed by routing is dependent on two major properties of the system: (i) the infrastructure (topology) of the communication network and (ii) the statistical pattern of route requests between sources and destinations. We first show that even in a very simple pattern such as every node has an activity level and the probability to send or receive a message is proportional to its level, the problem is NP-complete on general network topologies. Secondly, even when the network is "simpler" or regular, like a grid network, the problem can still be hard if the request distribution is "complex" in some sense. With this in mind we turn to analyze approximation algorithms for the setting where both the topology and the requests have nice properties. Our routing and activity distributions are partially justified from real data [8,9]. We concentrate on local and distributed algorithms, namely, processes can be exchanged (i.e., relocated) only between nearby nodes without any centralized coordination.

1.1. 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 in the network. When the network is a line, MEPL is identical to the minimum linear arrangement (MLA) [10] which is known to be NP-complete. In this work we consider d-dimensional grid networks, $d \geq 1$, and requests that come independently from a *symmetric product distribution* where the frequency of a route request (u, v) is a multiplication of the *activity* levels of both u and v. In contrast to previous works, our goal is to design simple distributed algorithms for these more realistic settings.

We first show that MEPL is NP-complete if (i): we only assume that the network is a 2-dimensional grid, and (ii): we only assume that the requests come from a *symmetric product distribution*. But, somewhat surprisingly, if both conditions hold, we are able to present a simple, local, distributed algorithm that achieves good approximation to the optimal solution for the MEPL problem. Our algorithm is self-adjustable in the sense that nodes switch processes based on the continuously observed sequence of route requests each node is involved in. This approach was inspired and bears some similarity to self-adjusting data structures like splay trees [11]. In particular we are able to show (informal):

Theorem. *For a* d-dimensional grid *network and a* symmetric product distribution *of requests there is a simple distributed algorithm, which defines a local switching policy between a process and its neighbors that 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. While for this setting few extremely unstable bad local minima can exist we present promising simulation results. In particular we show that for the 2-dimensional grid that starting from a random and thus almost worst case initial state of process locations in the network our local algorithms converge to an almost optimal local minimum.

Organization: In Section 2 we discuss related work and somewhat similar approaches. Section 3 introduces the formal problem and definitions. The hardness of MEPL is proved in Section 4 and then in Section 5 we prove our main result, a constant factor approximation in d-dimensional meshes with product route distributions. In Section 6 we discuss a more complex setting: clustered requests; and we end the paper with a short conclusion in Section 7.

2. Related work

Energy saving along with green computing is an active topic of research in the recent years. In a recent paper [12] Lis et al. study memory architectures of microprocessors. The authors suggest that processes will migrate to a location that is closer to the data instead of what is common in today's architectures, i.e., copying 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 before. Batista et al. [13] used traffic engineering on grids to self-adjust to routing requests. In [3] and [14] different authors considered data centers and tried to save energy by turning off routers and links when demand in the network is low. Other self-adjusting routing schemes were considered, e.g., in scale-free networks, to overcome congestion [15,16]. In [17], the authors studied virtual machines migration in a tree topology with the goal of minimizing the servers' load, while in [18], the problem of mapping processes to physical servers, in order to minimize congestion, was presented. In the current work, we optimize another important metric – average routing path length, which minimizes the number of lookups and transmissions performed by routers.

The most related areas of research to our study are graph *arrangement*, *embedding* and *labeling* problems [19,20]. The basic question is to embed a guest graph G into a host graph H in order to minimize some objective function like the *bandwidth* or the *cutwidth*; we relate our study to these settings in the model section. In particular, some VLSI design problems were considered on a two dimensional grid [21,22]. There are two significant differences here: first, we consider *distributions* on the route requests which restrict our guest graphs and second, and more important, we are interested in distributed, self-adjusting algorithms to solve the problem and not a centralized solution. In [23], the authors dealt with a problem that is similar to a special case of MEPL, where all nodes have the same activity level (uniform requests distribution), moreover, the proposed solution is centralized and not distributed.

As described in the introduction the self-adjusting nature of our solution was inspired by self-adjusting data structures like splay trees [11] which adjust their structure according to requests made to the data structure in such a way that the amortized cost matches the cost of the optimal (static) solution. Recently, Avin et al. in [24], extended the idea of splay trees to splay networks, i.e., self-adjusting trees that adapt themselves according to the pattern of routing requests to minimize the length of routing paths. Such a solution can be successfully used in overlay p2p networks.

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 [25] can be seen as simulating physics while cooling the temperature, i.e., the local moves shift, over time, more and more bias from mostly random behavior to greedy energy minimization. Here we only look at greedy steps. In a networking context similar approaches were used for load balancing via diffusive paradigms [26] and for routing via gradient mechanisms [27].

Another very related research is about self-stabilizing graphs [28,29]. The goal 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.

3. Model and problem definition

We model the communication network by an undirected, unweighted and connected *host* graph H. Given a graph, its vertex set is denoted $V(H)$ and its edge set by $E(H)$. We denote the number of nodes with $n = |V(H)|$ and the number of edges with $m = |E(H)|$. Let $\mathrm{d}_H(\cdot)$ be the distance function between nodes in H, i.e., for two nodes $u, v \in H$ we define $\mathrm{d}_H(u, v)$ to be the number of edges of a *shortest* path between u and v in H.

We assume that the network serves route requests drawn independently from an arbitrary distribution $\mathcal{P}$ and messages are routed along the shortest paths in H. Alternately, we represent the distribution $\mathcal{P}$ as a weighted directed *guest* graph G where $|V(G)| = n$. For a directed edge $(u, v) \in E(G)$ let the weight of the edge $\mathrm{p}(u, v)$ denote the probability of a route request for a message from node u to v.

Given a network infrastructure host graph H and a distribution on the route requests represented by a guest graph G, a ***placement*** (or *labeling* [19]) function is a bijective[1] function $\varphi : V(G) \to V(H)$ which determines the locations of nodes of G (processes) in the network H (hosts). Given G, H and a placement function φ, the expected path length of route requests is defined as:

$$\mathrm{EPL}(G, H, \varphi) = \sum_{(u,v)\in E(G)} \mathrm{p}(u, v) \cdot \mathrm{d}_H\big(\varphi(u), \varphi(v)\big)$$

When H and G are clear from the context we may write just $\mathrm{EPL}(\varphi)$. Note a special case of this definition, when $\mathcal{P}$ is the uniform distribution: this gives the *average path length* in the network which is often used in the literature instead of the diameter, for example to show that a network is a *small world network* [30].

For H and $\mathcal{P}$ we would like to find an optimal placement of the nodes in the network to minimize the expected path length. Formally:

Definition 1 *(Minimum expected path length problem).* Given a host graph H and a probability distribution represented by a guest graph G, find a placement function that minimize the expected path length:

$$\mathrm{MEPL} = \min_{\varphi} \mathrm{EPL}(G, H, \varphi)$$

[1] In this work we consider the classic case where every host machine can run at most one process (e.g., one virtual machine).

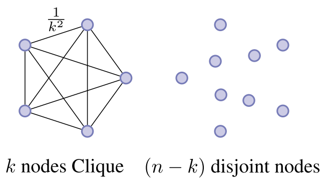

Fig. 1. Example of a symmetric product distribution. The guest graph consists of two parts: k nodes form a clique and the rest of the nodes are disjoint. Every node in the Clique has an activity level $\frac{1}{k}$, and all the other nodes have an activity level 0.

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 $\mathrm{d}(\varphi(u), \varphi(v))$ then the MEPL problem tries to minimize the expected cost of a route. Note that this is also equivalent to minimizing the expected number of lookups performed by routers, and minimizing the expected total number of transmissions – all important metrics in terms of energy saving and efficiency. More formally, if we denote by $\mathcal{E}(u, v)$ the energy spent by the network on serving request (u, v), then $\mathcal{E}(u, v) = (\mathcal{E}_{lookup} + \mathcal{E}_{trans})\mathrm{d}(\varphi(u), \varphi(v))$, where $\mathcal{E}_{lookup}$ and $\mathcal{E}_{trans}$ is the energy required by a router for a lookup and transmission operations respectively, and $\mathrm{d}(\varphi(u), \varphi(v))$ is the number of routers on the path (u, v) in H. By assuming that $\mathcal{E}_{lookup} + \mathcal{E}_{trans} = 1$ unit of energy, we obtain: $\mathcal{E}(u, v) = \mathrm{d}(\varphi(u), \varphi(v))$ and thus, the expectation of the total spent energy $\mathcal{E}$ is:

$$\begin{aligned}\mathbb{E}[\mathcal{E}] &= \sum_{(u,v)\in E(G)} \mathrm{p}(u, v) \cdot \mathcal{E}(u, v) \\ &= \sum_{(u,v)\in E(G)} \mathrm{p}(u, v) \cdot \mathrm{d}_H\big(\varphi(u), \varphi(v)\big) = \mathrm{EPL}(G, H, \varphi).\end{aligned}$$

In this work, we mostly consider local and distributed switching rules to find a good placement: rules where a process is only allowed to switch places with processes that are in its neighborhood (i.e., close 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. On the one hand, our results from Section 4 will show that this is not possible (efficiently) in a general setting even with global knowledge and non-local switches. Throughout the paper we thus consider at times simpler forms of networks and request distributions, i.e., *grid networks* and the *symmetric product distributions*:

Definition 2 *(d-Dimensional grid networks).* A d-dimensional grid is 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 ℓ_1-distance one from it, i.e., each node has at most two neighbors in each of the d dimensions.

Definition 3 *(Symmetric product distribution).* Let $\mathrm{p}(u)$ and $\mathrm{p}(v)$ be the activity levels of nodes u and v, respectively. If routing requests obey symmetric product distribution, then the probability of a routing request between u and v is given by $\mathrm{p}(u, v) = \mathrm{p}(u) \cdot \mathrm{p}(v)$.

In the other words, in symmetric product distribution, each node of G (i.e., process) 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. More precisely, we scale the activity levels of the nodes such that they form a distribution with an activity level $\mathrm{p}(u)$ for each node u and assume that the request distribution is induced by the product of the activity levels, i.e., $\mathrm{p}(u, v) = \mathrm{p}(u) \cdot \mathrm{p}(v)$. See Fig. 1 for an example of a guest graph that describes a symmetric product distribution.

In order to allow EPL optimization by means of the local switching rules, we make the following assumptions. First, in case of a *symmetric product distribution* of requests, a node (process) u can measure the activity level of any other node v by simply calculating the ratio between the frequency of the requests (u, v) and sum of the frequencies of all the other requests (u, w), where $w \in V$. Second, in order to be able to make EPL calculation locally, a node needs to know the locations of all other nodes in H. Thus, we assume the existence of some central directory service, similar to a DNS service, that keeps track of the nodes' locations (addresses). Note that if the request distribution is not *symmetric product*, activity levels cannot be measured locally, and, in this case, an additional centralized directory may be used to track the nodes' activity.

4. Hardness of MEPL

In this section we show that solving the general MEPL problem is hard. Indeed, we prove two results that demonstrate how the hardness of the problem can come from either an involved network topology 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 a more realistic structure.

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

Lemma 1. *Given a host graph H and a symmetric product distribution of requests $\mathcal{P}$, it is NP-complete to decide whether the MEPL is smaller than a given value.*

Proof. We describe a reduction from the k-CLIQUE problem. In the k-CLIQUE problem, one is given a graph H' and has to decide whether H' contains a k-clique, that is, whether H' contains a complete graph on k nodes as a subgraph. This is one of Karp's 21 NP-complete problems [31]. For the reduction we take the graph H' as the network's host graph H. As a request distribution we use a symmetric product probability distribution that puts $1/k^2$ 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 (this is the distribution that was used as an example in Fig. 1). If H contains a k-clique, then the unique optimal solution to MEPL with value $k(k-1)/k^2 = 1 - 1/k$ will be obtained if all k nodes are placed in this clique. If H does not contain a k-clique then there will be at least one request pair $u, v \in V' \times V'$ that is at least two hops apart and the total cost will be at least $1 - 1/k + 1/k^2$. Thus, deciding whether the MEPL is smaller than $1 - 1/k + 1/k^2$ is equivalent to deciding whether H' has a k-CLIQUE and thus, NP-hard. Lastly, it is easy to see that deciding whether the MEPL is smaller than a given value problem can be easily achieved in NP by guessing and then verifying a solution with smaller value. □

This lemma shows that solving the MEPL problem for general network topologies is hard. Next, we show that even if we restrict the graph to be nice, e.g., the 2-dimensional grid, a lack of structure in the probability distribution can make the MEPL problem hard, too:

Lemma 2. *Given a probability distribution of requests $\mathcal{P}$, it is NP-complete to decide whether the MEPL is smaller than a given value on a 2-dimensional grid network.*

Proof. We describe a reduction from 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 T (with maximum degree 4) is a subgraph of the grid. Given an instance of this problem in form of a tree T we construct a hard MEPL instance as follows: We take the two-dimensional grid $[k]^2$ as a topology where k equals the number of nodes in T. As a request distribution we take a subset of k nodes to correspond to nodes in T and put a probability mass of $1/(k-1)$ on each pair of nodes that corresponds to two neighbors in the tree T; all other $k^2 - (k-1)$ node pairs have a probability of zero. The MELP for such an instance is 1 if and only if the tree can be embedded in the grid. If this is not the case then at least one request pair will be separated by a path of length at least two increasing the average to at least $1 + 1/(k-1)$. Thus deciding whether the MEPL is smaller than $1 + 1/(k-1)$ is equivalent to deciding whether T can be embedded into the 2-dimensional grid. This proves solving the MEPL problem on the 2-dimensional grid NP-hard. □

Contrasting these two hardness results, the next sections will show that if one assumes a grid graph *and* a symmetric product distribution, nice algorithmic results can be obtained.

5. Distributed MEPL with symmetric product distributions

For a 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. With this in mind, we first restrict ourselves to a simpler model of requests, namely, symmetric product distributions (Definition 3). Second, we assume d-dimensional grid topologies, and in particular the line and a 2-dimensional grid. We assume that a process learns the distribution from requests it is 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 simple (greedy) local switching rules. To prove this result, we need the following definitions.

5.1. Expected distance to center and expected rank

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

Definition 4 *(Center and expected distance to the center).* Consider a symmetric product distribution $\mathcal{P}$ (with its corresponding guest graph G) and a host graph H. Then, for a placement φ, the expected center of H is a node c^*, s.t.:

$$c^*(G, H, \varphi) = \arg\min_{x \in V(H)} \sum_{u \in V(G)} \mathrm{p}(u)\mathrm{d}\big(\varphi(u), x\big).$$

The expected distance to the center for φ, $c^*(\varphi)$ is:

$$C(G,H,\varphi) = \min_{x\in V(H)} \sum_{u\in V(G)} \mathrm{p}(u)\mathrm{d}\big(\varphi(u),x\big),$$

or equally:

$$C(G,H,\varphi) = \sum_{u\in V(G)} \mathrm{p}(u)\mathrm{d}\big(\varphi(u),c^*\big).$$

When H and $\mathcal{P}$ are clear from the context (recall that $\mathcal{P}$ defines the guest graph G), both C and c^* can be written simply as $C(\varphi)$ and $c^*(\varphi)$. The minimum expected distance to the center is defined then as $C_{\min} = \min_\varphi C(\varphi)$. The next lemma describes the relation between C and EPL.

Lemma 3. *Consider a symmetric product distribution $\mathcal{P}$ and a host graph H. For any given placement φ, $2C(\varphi) \geq \mathrm{EPL}(\varphi) \geq C(\varphi)$.*

Proof. To see the upper bound, we suppose that instead of routing between two nodes directly, we route every request via the center 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:

$$\begin{aligned}
\mathrm{EPL}(\varphi) &= \sum_{(u,v)\in E(G)} \mathrm{p}(u)\mathrm{p}(v)\cdot \mathrm{d}\big(\varphi(u),\varphi(v)\big)\\
&\leq \sum_{(u,v)\in E(G)} \mathrm{p}(u)\mathrm{p}(v)\big(\mathrm{d}\big(\varphi(u),c^*\big)+\mathrm{d}\big(c^*,\varphi(v)\big)\big)\\
&= \sum_{u\in V(G)} \mathrm{p}(u)\mathrm{d}\big(\varphi(u),c^*\big) + \sum_{v\in V(G)} \mathrm{p}(v)\mathrm{d}\big(c^*,\varphi(v)\big) = 2C(\varphi)
\end{aligned}$$

The fact that C is a lower bound is shown as follows:

$$\begin{aligned}
\mathrm{EPL}(\varphi) &= \sum_{(u,v)\in E(G)} \mathrm{p}(u)\mathrm{p}(v)\cdot \mathrm{d}\big(\varphi(u),\varphi(v)\big)\\
&= \sum_{u\in V(G)} \mathrm{p}(u) \sum_{v\in V(G)} \mathrm{p}(v)\mathrm{d}\big(\varphi(u),\varphi(v)\big)\\
&\geq \sum_{u\in V(G)} \mathrm{p}(u) \sum_{v\in V(G)} \mathrm{p}(v)\mathrm{d}\big(c^*,\varphi(v)\big)\\
&= \sum_{v\in V(G)} \mathrm{p}(v)\mathrm{d}\big(c^*,\varphi(v)\big) = C(\varphi) \qquad \square
\end{aligned}$$

Corollary 1. $\mathrm{MEPL} \geq C_{\min}$

This follows since for the optimal placement φ^*: $\mathrm{MEPL} = \mathrm{EPL}(\varphi^*) \geq C(\varphi^*) \geq C_{\min}$.

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 5 *(Rank of nodes and the expected rank).* The rank of a node is the position of the node in the ordered list of the 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 $\mathrm{r}(u)$. The expected rank of a probability distribution on the nodes of graph G is: $\mathbb{E}[R] = \sum_{u\in V(G)} \mathrm{p}(u)\mathrm{r}(u)$.

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

5.2. (Greedy) local switching strategies

For two nodes $u, v \in V(G)$ and a placement φ, we say that u is a *neighbor* of v if and only if $(\varphi(u)\varphi(v)) \in E(H)$. A switching of u and v is taken to be understood as a new placement φ' where for each $w \in V(G)$, $w \neq u, v$ $\varphi'(w) = \varphi(w)$ and $\varphi'(u) = \varphi(v)$ and $\varphi'(v) = \varphi(u)$, i.e., u and v switch places on H.

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**: A 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**: A node will switch locations with its neighbor 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 when this decreases the expected path length (or some other criterion), then it is clear that this, strictly monotone, potential cannot drop indefinitely (or too often) and thus, a (quick) convergence is guaranteed. A placement φ is said to be a ***local minimum*** (or local optimum) if and only if 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 the local minimum placement. First, few definitions are in place.

Definition 6 *(Center distance decreasing path).* Let φ be a placement and $\varphi(u)$ and $\varphi(v)$ be the positions of the processes u and v, respectively. Consider a path $P = (\varphi(u) = w_1, w_2, \ldots, w_k = \varphi(v))$ between $\varphi(u)$ and $\varphi(v)$. The path P is a *φ-center distance decreasing path* if for each $i \in [1, \ldots, k-1]$, $\mathrm{d}(w_i, c^*(\varphi)) > \mathrm{d}(w_{i+1}, c^*(\varphi))$ where $c^*(\varphi)$ is the center of φ.

Definition 7 *(Center monotone placement).* Center monotone placement is a placement φ for which the following holds. For any $u, v \in G$, if there is a center distance decreasing path from $\varphi(u)$ to $\varphi(v)$, then $\mathrm{p}(u) \le \mathrm{p}(v)$.

Lemma 4. *Any local minimum placement φ with respect to the C-rule, is* center monotone.

Proof. Assume for the sake of contradiction that φ is a local minimum, that $\mathrm{p}(u) > \mathrm{p}(v)$ and that there is a *center distance decreasing path P* from $\varphi(u)$ to $\varphi(v)$ (see Definition 6). Since $\mathrm{p}(u) > \mathrm{p}(v)$, there has to be two nodes u' and v' such that $\varphi(u')$ and $\varphi(v')$ are neighbors on the path P with $\mathrm{p}(u') > \mathrm{p}(v')$ but $\varphi(v')$ is closer to $c^*(\varphi)$ than $\varphi(u')$. It is possible to switch $\varphi(u')$ and $\varphi(v')$ and it is easy to see that this is an improvement with regards to the C-rule. This contradicts the assumption that φ is a local minimum. □

Note that according to the C-rule, two neighbors switch locations 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 $c^*(\varphi)$ can be computed locally at each node via the activity levels (due to the product distribution, each node sees the same activity levels of all the other nodes) and the location information of all the nodes. Therefore, the C-rule will greedily minimize, for each node, the distance to the expected position of its requests.

In order to implement our local and distributed switching strategies, there is a need for a protocol, in which at each step (synchronous or asynchronous) a node will choose a neighbor to negotiate the switching. Additionally, all the nodes should know the locations and activity levels of all the other nodes. As already discussed in Section 3, the location information can be provided using a centralized directory service, and the activity level, in case of a *symmetric product distribution*, can be obtained locally by just counting the number of requests to and from specific peer. Detailed description of such protocol is out of the scope of the current work which deals with the performance analysis of the local and distributed switching strategies compared to a global optimal nodes placement.

Comment As pointed out by an anonymous reviewer, Lemma 4 holds also for any arbitrary constant center location c^* (and not only for the dynamic, $c^*(\varphi)$, center). The constant center makes the distributed nature of the algorithm clearer, since there no need for a node to know the activity levels of all the nodes and their positions. However, starting the process (convergence) from the current center $c^*(\varphi)$ (with minimum expected path length), and not from an arbitrary constant center, will expedite the convergence time.

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.

5.3. 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 H be the line and $\mathcal{P}$ a symmetric product distribution, then any locally optimal solution achieved by the C-rule (or M-rule) is at most a factor of four larger than the global optimum of MEPL.*

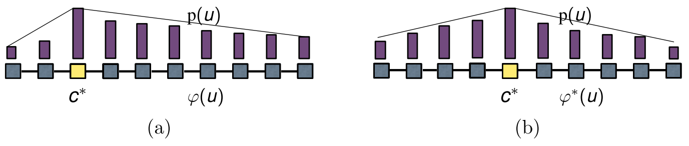

Fig. 2. (a) – Local minimum placement φ on the line host graph H. For any local minimum we have: $\mathrm{d}(\varphi(u), c^*) \le \mathrm{r}(u)$ for all $u \in V(G)$. (b) – Global minimum placement φ^* on the line host graph H. For any global minimum we have: $\mathrm{d}(\varphi^*(u), c^*) \ge \mathrm{r}(u)/2$ for all processes $u \in V(G)$.

We prove this theorem for the C-rule but this could be done similarly for the M-rule. Assume H and $\mathcal{P}$ 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 φ achieved by the C-rule: $C(\varphi) \le \mathbb{E}[R]$, and $\mathrm{EPL}(\varphi) \le 2\mathbb{E}[R]$.*

Proof. Let $\mathrm{d}(\varphi(v), c^*)$ be the distance of $\varphi(v)$ from c^*. We want to bound it in terms of $\mathrm{r}(v)$, the rank of v. From Lemma 4 we get that all nodes between $\varphi(v)$ and c^* on the line have higher probability than v and thus $\mathrm{d}(\varphi(v), c^*) \le \mathrm{r}(v)$. So: $C(\varphi) = \sum_{v\in V(G)} \mathrm{p}(v)\mathrm{d}(\varphi(v), c^*) \le \sum_{v\in V(G)} \mathrm{p}(v)\mathrm{r}(v) = \mathbb{E}[R]$. From Lemma 3, $\mathrm{EPL}(\varphi) \le 2C(\varphi) \le 2\mathbb{E}[R]$. □

We now prove a lower bound for MEPL on the line and any symmetric product distribution of requests.

Lemma 6. $\mathrm{MEPL} \ge C_{\min} \ge \frac{1}{2}\mathbb{E}[R]$.

Proof. Let φ^* be the placement such that $C_{\min} = C(\varphi^*)$. Note that by definition, φ^* minimizes the expected path length to the center. Given the center $c^*(\varphi^*)$ and an arbitrary node v with a distance $\mathrm{d}(\varphi^*(v), c^*)$ from c^*, we want to find an upper bound on the rank of v by bounding how many nodes can have a higher activity level than v. Clearly, all such nodes will be at most at the distance $\mathrm{d}(\varphi^*(v), c^*)$ from the center, since otherwise, φ^* will not be optimal. Since in a line there at most two nodes at distance i from the center $\mathrm{d}(\varphi^*(v), c^*) \ge \mathrm{r}(v)/2$ we obtain as desired: $C_{\min} = \sum_{v\in V} \mathrm{p}(v)\mathrm{d}(\varphi^*(v), c^*) \ge \sum_v \mathrm{p}(v)\mathrm{r}(v)/2 = \mathbb{E}[R/2]$. □

For a graphical illustration of Lemmas 5 and 6, see Fig. 2. To conclude the proof of Theorem 1, we combine Lemmas 3, 5 and 6 to get that for a local minimum φ: $2\mathbb{E}[R] \ge \mathrm{EPL}(\varphi) \ge \mathrm{MEPL} \ge \frac{1}{2}\mathbb{E}[R]$. Thus, the ratio between the worst case local solution and the optimal solution is at most 4.

5.4. The d-dimensional grid

In this section, we extend the ideas from the line to grid networks. Our results apply readily to grids of arbitrary dimension but, for the sake of simplicity, we stick to two dimensions here. We first show that using the same greedy approach as in the line, namely switching neighboring nodes using the M-rule, leads to a drastically worse ratio between local and global minima.

Lemma 7. *On the d-dimensional grid, there is a local minimum with regards to the M-rule and the C-rule that is a factor of $\Omega(n^{1/d-1/d^2})$ worse than the global minimum.*

Proof. We take $n^{1/d}$ nodes with uniform probability $p = n^{-1/d}$ and set all other nodes to probability zero. We arrange the active nodes on a line along one dimension as an initial placement. We now note that this initial placement is a local minimum since all switches are non-improving with regards to the M-rule or C-rule and also lead to a higher expected path length. This is true because any switch between nodes in the line does not change the expected path length while switching a node from the line with an inactive node only increases the path length for the active node by one while the reduced path length for the inactive node has no effect on the expected path length. The same is true for the expected distance to the center. The expected path length of this linear arrangement is $n^{1/d}/2$. To complete the proof we will now argue that there are arrangements with EPL of order $O(n^{1/d^2})$. One such (close to optimal) solution can be achieved by arranging all active nodes within a ball around the center (see Fig. 3(b) for an illustration of such an arrangement for $d = 2$). Since the number of nodes in a ball in d dimensions grows as r^d with the radius d it is clear that a radius of $r = O(n^{1/d^2})$ suffices to contain enough spots to place the $n^{1/d}$ active nodes in. The maximum (and also expected) path length within the ball is also within a constant factor of the radius, i.e., also $O(n^{1/d^2})$. The ratio between the local minimum and the alternative ball arrangement is thus $\Omega(n^{1/d-1/d^2})$ proving that the ratio between the worst local minimum compared to the global minimum is at least this large. □

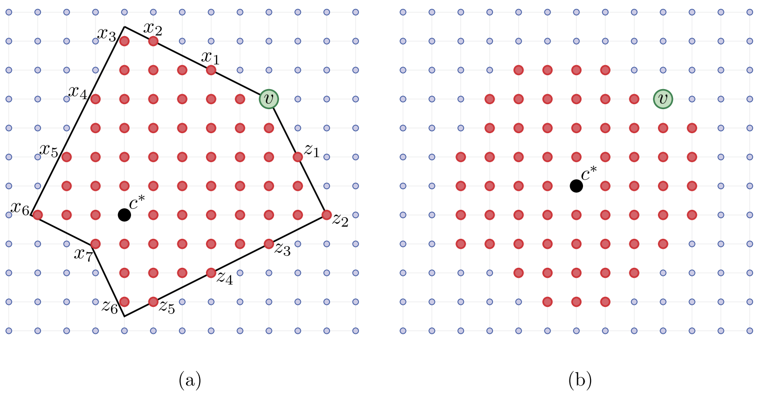

Fig. 3. (a) – Local minimum placement φ on the 2-d grid host graph H. For any local minimum we have: $\mathrm{d}(\varphi(v), c^*) \le \frac{4}{\sqrt{6}}\mathrm{r}(v)$ for all $v \in V(G)$. (b) – Global minimum placement φ^* on the 2-d grid host graph H. For any global minimum we have: $\mathrm{d}(\varphi^*(v), c^*) \ge \sqrt{\frac{\mathrm{r}(v)}{2}}$ for all processes $v \in V(G)$.

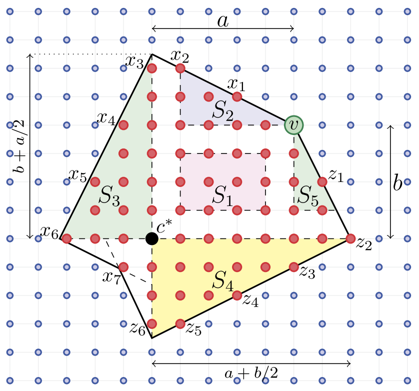

Fig. 4. A local minimum when optimizing the distance to the expected position of a routing request (black node c^*). The red nodes mark the positions that dominate the green node v, that is, that have a path consisting of possible switchings each goes strictly closer to the black node. In a local minimum these nodes have to have a larger probability than the green node making the rank of the green node at least as large as the number of red nodes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Note that the last lemma implies an $\Omega(n^{1/4})$ worst-case ratio for the 2-dimensional grid. Surprisingly, we can avoid this, locally stable but highly suboptimal solution, by allowing only slightly longer switches. The rule we propose is that a node can also switch with any of the neighbors in ℓ_1-distance three that differs two in one axis and one 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 H be the 2-dimensional grid and $\mathcal{P}$ a symmetric product distribution, then any locally optimal solution achieved by the C-rule (and allowing "chess 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, where we provide bounds on C for the optimal and locally optimal placements. Fig. 3 illustrates the local and the global minimum cases which are formally analyzed in Lemmas 8 and 9. First, we show that we get a fat set from this strategy and also prove a stronger rank-property of any locally optimal solution; namely, nodes that are far away from the center have to have a (quadratically) high rank.

Lemma 8. *For any local minimum of the C-rule $C(\varphi) \le \frac{4}{\sqrt{6}}\mathbb{E}[\sqrt{R}]$, where R is the rank of a given distribution (see Definition 5).*

Proof. Consider a node v in general relative position to the center c^* (see Fig. 4). We want to estimate how many nodes have a larger probability (higher rank) than the node v. To achieve this estimation, we analyze the area of the largest polygon, such that every node inside and on the edges of the polygon belongs to some *center distance decreasing path* from v to c^* (see Definition 6). According to Lemma 4, we get the guarantee that for any local minimum, with respect to the C-rule, any node u in the polygon has $\mathrm{p}(v) \le \mathrm{p}(u)$. Fig. 4 provides an example for this: The node x_1 is closer to the center than v and switching between x_1 and v is possible. This implies that $\mathrm{p}(v) \le \mathrm{p}(x_1)$. Furthermore, if we look at the *center distance decreasing paths* $v, x_1, x_2, \dots, x_7$ and $v, z_1, z_2, \dots, x_7$ we obtain from Lemma 4 that $\mathrm{p}(v) \le \mathrm{p}(x_1) \le \mathrm{p}(x_2) \le \dots \le \mathrm{p}(x_7)$ and that $\mathrm{p}(v) \le \mathrm{p}(z_1) \le \dots \le \mathrm{p}(x_7)$. These paths can furthermore be extended to any node in the polygon. All these nodes have higher activity levels then $\mathrm{p}(v)$. To get a bound on the rank of v, i.e., on the number of nodes that have a larger activity level, we count the number of nodes that are inside the polygon. This number generally involves many floors and ceilings. We avoid these by first calculating the number of nodes $A(x)$ bounded by a right triangle shape that starts at a point and whose (axis parallel) legs have length x and $x/2$. We denote this number by $A(x)$ and in the following give a formula and an estimate for it that holds for any positive real x:

$$A(x) := \sum_{i=0}^{\lfloor x/2 \rfloor} \left(\lfloor x \rfloor + 1 - 2i\right) \geq \begin{cases} \frac{x^2}{4} + 1 & \text{if } x \geq 1, \\ 1 & \text{if } x < 1. \end{cases}$$

Now we are ready to calculate 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 Fig. 4.

$$S_1 = (a-1)(b-1) \qquad S_2 = A(a-1) \qquad S_3 = A(b+a/2)$$
$$S_4 = A(a+b/2) \qquad S_5 = A(b-1)$$

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 for $a, b \geq 2$: $S_{total} = S_1 + S_2 + S_3 + S_4 + S_5 - 3 \geq \frac{6}{16}(a+b)^2$. It is easy to verify that the inequality $S_{total} \geq \frac{6}{16}(a+b)^2$ holds also in the case where a, or b, or both are equal to 1. Since the rank of v is at least S_{total}, we get $\mathrm{r}(v) \geq \frac{6}{16}(\mathrm{d}(v, c^*))^2$, and thus: $\mathrm{d}(v, c^*) \leq \frac{4}{\sqrt{6}}\sqrt{\mathrm{r}(v)}$. Hence: $C = \sum_{v \in V} \mathrm{p}(v)\mathrm{d}(v, c^*) \leq \sum_{v \in V} \mathrm{p}(v) \frac{4}{\sqrt{6}}\sqrt{\mathrm{r}(v)} = \frac{4}{\sqrt{6}}\mathbb{E}[\sqrt{R}]$. □

Next we show a lower bound for the cost of the optimum placement obtaining a similar expression in terms of the (expected) rank.

Lemma 9. MEPL $\geq C_{\min} \geq \frac{1}{\sqrt{2}}\mathbb{E}[\sqrt{R}]$.

Proof. Let φ^* be the placement such that $C_{\min} = C(\varphi^*)$. Note that by definition φ^* minimizes the expected path to the center. Given the center $c^*(\varphi^*)$ and an arbitrary node v with a distance $\mathrm{d}(v, c^*)$ from c^*, we again find an upper bound on the rank of v by showing how many nodes can have a larger activity level than v. Again, all such nodes will be at most at the distance $\mathrm{d}(v, c^*)$ from the center, since otherwise the solution will not be a global optimum. There are exactly 4 nodes at distance 1 from c^*, 8 nodes at the distance 2 and in general $4i$ nodes at distance i. This leads to the following bound: $\mathrm{r}(v) \leq \sum_{i=1}^{\mathrm{d}(v,c^*)} 4i = 2(\mathrm{d}(v, c^*))^2$. So, we obtain $\mathrm{d}(v, c^*) \geq \sqrt{\mathrm{r}(v)/2}$, and thus: $C = \sum_{v \in V} \mathrm{p}(v)\mathrm{d}(v, c^*) \geq \sum_v \mathrm{p}(v)\sqrt{\mathrm{r}(v)/2} = \mathbb{E}[\sqrt{R/2}]$. □

Now we are ready to prove the result of Theorem 2. From Lemmas 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 $C(\varphi) \leq \mathbb{E}[\frac{4}{\sqrt{6}}\sqrt{R}]$, and thus by Lemma 3 the expected path length of a local minimum cannot be larger than $2\mathbb{E}[\frac{4}{\sqrt{6}}\sqrt{R}]$. Therefore the ratio between the optimal solution and any local minimum with regards to the C-rule is at most $2\frac{4\sqrt{2}}{\sqrt{6}} \approx 4.62$.

All proofs above can be extended to the d-dimensional grid. The lower bound guarantees in this context that any solution on the d-dimensional grid has a cost of at least $\Omega(E[R^{1/d}])$, where R is the rank of a sampled node. Similarly for any constant d we get a $O(E[R^{1/d}])$ upper bound from the fact that a fat body is dominated by any node. The constant factor in the upper bound decreases like 2^{-d} since using the *chess knight* improvement switches along any dimension costs a factor of 2. By using longer improvement directions of length k instead of 3 the factor of two can be brought down to $1 - \frac{1}{k-2}$. Thus, e.g., using length d improvement switches on the d-dimensional grid results in a constant factor approximation for any dimension d.

6. Clustered requests

We have demonstrated that self-adjusting networks and their local switching rules work well on grid networks with symmetric product distributions. We now briefly discuss some interesting preliminary results on a more general type of request distributions: clustered requests. For this we consider situations where processes can be clustered into groups such that communication takes place only or predominantly between processes belonging to the same group. This locality is inspired by practice and we believe that such a structure in the requests is quite common.

Ideally, a self-adjusting network “detects” such clusters and arranges the processes such that groups will reside in separate parts of the network infrastructure. Such an arrangement facilitates short routes since requests between group members get routed quickly without leaving the group. Such a well clustered placement of nodes can be a drastic improvement of the expected path length compared to a non-optimized placement. In particular, any placement that is oblivious to the

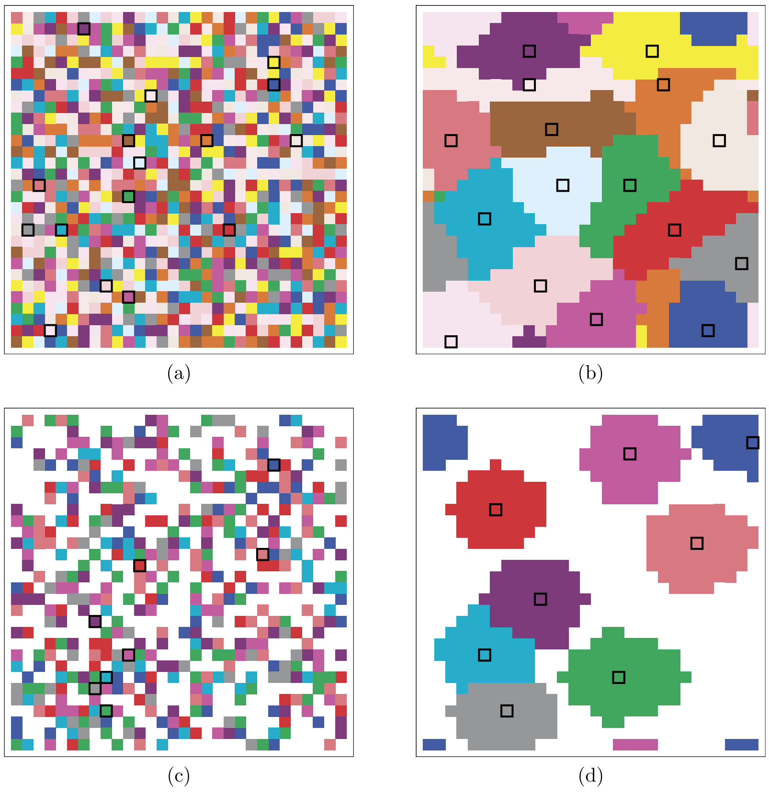

Fig. 5. (a) Torus with 900 nodes, 16 clusters and no nodes in the inactive cluster. Nodes are placed at random positions. (b) Final placement after applying the local greedy strategy in a round robin fashion. Clusters are grouped together, but their shapes are not optimal for a given cluster. (c) Torus with 900 nodes, 8 active clusters and half of the nodes in the inactive cluster. V_0 has $n/2$ nodes. Nodes are placed at random positions. (d) Final placement after applying the local greedy strategy. Clusters are grouped together and almost optimally shaped around their center. (For interpretation of the colors in this figure, the reader is referred to the web version of this article.)

clustering, e.g., a random placement, will have a bad performance – leaving plenty of room for improvement. We believe that the simple switching strategies presented in this paper perform very well in many such settings.

Our investigations and simulations on d-dimensional torus topologies have led to several interesting preliminary results in this direction: On the negative side we were able to construct local minima in many topologies that have a poor performance compared to the global optimum. This shows that it is not possible to give the same type of strong approximation guarantee, independent of the initialization. In simulations, we observed that for $d = 1$, i.e., on a ring topology, even random initializations lead to bad local minima. The reason for this is that connectivities in the ring are too restricted to allow the resolution of distributed clusters without disturbing other, already fixed, clusters. Fortunately, this changes drastically for higher dimensions. For $d > 1$ local minima become extremely unstable and sensitive to small perturbations. They can only occur if one starts in a carefully constructed worst-case initialization. This observation is supported by our simulations which produce very promising results. Fig. 5 shows an example for the improvement in a 2-dimensional torus. Overall, the greedy switching algorithm successfully detects clusters and groups them together. However, there are still some clusters that are not connected. In Fig. 5(a) we can see the initial random placement of the clustered nodes. Nodes with a black bold frame 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 Fig. 5(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 Fig. 5(c) we can see the initial random placement of the clustered nodes, (the inactive nodes are white colored). In Fig. 5(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. Animated version of the figures can be found here: http://www.bgu.ac.il/~avin/pmwiki/pmwiki.php?n=Main.Self-AdjustingNetworks.

7. Conclusions and future work

In this preliminary work, we formally defined the MEPL problem which has practical significance in saving energy of fixed infrastructure network. We showed that in general cases, the problem is hard to compute, but under some realistic assumptions on network infrastructure and traffic patterns, we propose efficient local and distributed algorithms that achieve almost optimal solution. Our algorithms are based on migration of processes, which allows network optimization without changing the underlying infrastructure. This idea integrates well with an SDN concept which will probably include process migration functionality in its management platform.

In future work, we plan to extend our results to other topologies that are used in data center networks, e.g., fat trees [32]. We also aim to investigate other types of requests distributions that are based on real data.

Acknowledgement

We would like to thank the anonymous reviewers who helped us to significantly improve the paper.

References

[1] M. Poess, R. Nambiar, Energy cost, the key challenge of today's data centers: a power consumption analysis of tpc-c results, in: Proceedings of the VLDB Endowment, vol. 1(2), 2008, pp. 1229–1240.
[2] U.S. Environmental Protection Agency, Report to congress on server and data center energy efficiency public law 109–431, 2007.
[3] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, N. McKeown, Elastictree: saving energy in data center networks, in: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, USENIX Association, 2010, p. 17.
[4] M. Mirza-Aghatabar, S. Koohi, S. Hessabi, M. Pedram, An empirical investigation of mesh and torus NoC topologies under different routing algorithms and traffic models, in: 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, DSD 2007, IEEE, 2007, pp. 19–26.
[5] K. Greene, TR10: software-defined networking, http://www.technologyreview.com/web/22120/.
[6] U. Hoelzle, Openflow @ google, Open Networking Summit, 2012.
[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, Openflow: enabling innovation in campus networks, SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 69–74, http://dx.doi.org/10.1145/1355734.1355746.
[8] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, J. Zahorjan, Measurement, modeling, and analysis of a peer-to-peer file-sharing workload, in: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, ACM, 2003, pp. 314–329.
[9] A. Klemm, C. Lindemann, M. Vernon, O. Waldhorst, Characterizing the query behavior in peer-to-peer file sharing systems, in: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement, ACM, 2004, pp. 55–67.
[10] D. Johnson, M. Garey, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman&Co, San Francisco, 1990.
[11] D. Sleator, R. Tarjan, Self-adjusting binary search trees, J. ACM 32 (3) (1985) 652–686.
[12] M. Lis, K. Shim, M. Cho, C. Fletcher, M. Kinsy, I. Lebedev, O. Khan, S. Devadas, Brief announcement: distributed shared memory based on computation migration, in: SPAA, ACM, 2011, pp. 253–256.
[13] D. Batista, N. da Fonseca, F. Granelli, D. Kliazovich, Self-adjusting grid networks, in: IEEE International Conference on Communications, 2007, ICC'07, IEEE, 2007, pp. 344–349.
[14] Y. Shang, D. Li, M. Xu, Energy-aware routing in data center network, in: Proceedings of the First ACM SIGCOMM Workshop on Green Networking, Green Networking '10, ACM, New York, NY, USA, 2010, pp. 1–8, http://doi.acm.org/10.1145/1851290.1851292.
[15] M. Tang, Z. Liu, X. Liang, P.M. Hui, Self-adjusting routing schemes for time-varying traffic in scale-free networks, Phys. Rev. E 80 (2) (2009) 026114, http://dx.doi.org/10.1103/PhysRevE.80.026114.
[16] H. Zhang, Z. Liu, M. Tang, P. Hui, An adaptive routing strategy for packet delivery in complex networks, Phys. Lett. A 364 (3–4) (2007) 177–182, http://dx.doi.org/10.1016/j.physleta.2006.12.009.
[17] N. Jain, I. Menache, J.S. Naor, F.B. Shepherd, Topology-aware VM migration in bandwidth oversubscribed datacenter networks, in: Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming – Volume Part II, ICALP'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 586–597, http://dx.doi.org/10.1007/978-3-642-31585-5_52.
[18] N. Bansal, K.-W. Lee, V. Nagarajan, M. Zafer, Minimum congestion mapping in a cloud, in: Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC '11, ACM, New York, NY, USA, 2011, pp. 267–276, http://doi.acm.org/10.1145/1993806.1993854.
[19] F. Chung, Labelings of graphs, in: Selected Topics in Graph Theory, vol. 3, 1988, pp. 151–168.
[20] J. Díaz, J. Petit, M. Serna, A survey of graph layout problems, ACM Comput. Surv. 34 (2002) 313–356, http://dx.doi.org/10.1145/568522.568523.
[21] S. Bhatt, S. Cosmadakis, The complexity of minimizing wire lengths in VLSI layouts, Inform. Process. Lett. 25 (4) (1987) 263–267.
[22] S. Bhatt, F. Thomson Leighton, A framework for solving VLSI graph layout problems, J. Comput. System Sci. 28 (2) (1984) 300–343.
[23] E.D. Demaine, S.P. Fekete, G. Rote, N. Schweer, D. Schymura, M. Zelke, Integer point sets minimizing average pairwise l1 distance: what is the optimal shape of a town?, Comput. Geom. Theory Appl. 44 (2) (2011) 82–94, http://dx.doi.org/10.1016/j.comgeo.2010.09.004.
[24] C. Avin, B. Haeupler, C. Scheideler, S. Schmid, Locally self-adjusting tree networks, in: 27th IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2013.
[25] S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671.
[26] Y. Rabani, A. Sinclair, R. Wanka, Local divergence of Markov chains and the analysis of iterative load-balancing schemes, in: Focs, IEEE Computer Society, 1998, pp. 694–703.
[27] S. Mukherjee, N. Gupte, Gradient mechanism in a communication network, Phys. Rev. E 77 (3) (2008) 036121, http://dx.doi.org/10.1103/PhysRevE.77.036121.
[28] R. Jacob, A. Richa, C. Scheideler, S. Schmid, H. Täubig, A distributed polylogarithmic time algorithm for self-stabilizing skip graphs, in: Proceedings of the 28th ACM Symposium on Principles of Distributed Computing, ACM, 2009, pp. 131–140.
[29] R. Jacob, S. Ritscher, C. Scheideler, S. Schmid, A self-stabilizing and local Delaunay graph construction, in: Algorithms and Computation, 2009, pp. 771–780.
[30] D. Watts, S. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 440–442.
[31] R. Karp, Reducibility among combinational problems, in: Complexity of Computer Computations, 1972, pp. 85–104.
[32] C.E. Leiserson, Fat-trees: universal networks for hardware-efficient supercomputing, IEEE Trans. Comput. 34 (10) (1985) 892–901.