Brief Announcement: Self-Adjusting Linear Networks

Chen Avin[1] Ingo van Duijn[2] Stefan Schmid[3]

[1] Ben Gurion University of the Negev
[2] Aalborg University
[3] University of Vienna

Abstract. Emerging networked systems become increasingly flexible and "reconfigurable". This introduces an opportunity to adjust networked systems in a demand-aware manner, leveraging spatial and temporal locality in the workload for *online* optimizations. However, it also introduces a tradeoff: while more frequent adjustments can improve performance, they also entail higher reconfiguration costs.

This paper initiates the formal study of *linear* networks which self-adjust to the demand in an online manner, striking a balance between the benefits and costs of reconfigurations. We show that the underlying algorithmic problem can be seen as a distributed generalization of the classic dynamic list update problem known from self-adjusting datastructures: in a network, requests can occur between *node pairs*. This distributed version turns out to be significantly harder than the classical problem in generalizes. Our main results are a $\Omega(\log n)$ lower bound on the competitive ratio, and a (distributed) online algorithm that is $\mathcal{O}(\log n)$-competitive if the communication requests are issued according to a linear order.

Keywords: Self-adjusting datastructures · competitive analysis · distributed algorithms · communication networks.

1 Introduction

Communication networks are becoming increasingly flexible, along three main dimensions: routing (enabler: software-defined networking), embedding (enabler: virtualization), and topology (enabler: reconfigurable optical technologies, for example [3]). In particular, the possibility to quickly reconfigure communication networks, e.g., by migrating (virtualized) communication endpoints [1] or by reconfiguring the (optical) topology [2], allows these networks to become *demand-aware*: i.e., to adapt to the traffic pattern they serve, in an online and *self-adjusting* manner. For example, in a self-adjusting network, frequently communicating node pairs can be moved *topologically closer*, saving communication costs (e.g., bandwidth, energy) and improving performance (e.g., latency, throughput).

However, today, we still do not have a good understanding yet of the algorithmic problems underlying self-adjusting networks. The design of such algorithms faces several challenges. As the demand is often not known ahead of time, *online* algorithms are required to react to changes in the workload in a clever way; ideally, such online algorithms are "competitive" even when compared to an optimal offline algorithm which knows the demand ahead of time. Furthermore, online algorithms need to strike a balance between the benefits of adjustments (i.e., improved performance and/or reduced costs) and their costs (i.e., frequent adjustments can temporarily harm consistency and/or performance, or come at energy costs).

The vision of self-adjusting networks is reminiscent of self-adjusting datastructures such as *self-adjusting lists* and *splay trees*, which optimize themselves toward the workload. In particular, the *dynamic list update problem*, introduced already in the 1980s by Sleator and Tarjan in their seminal work [4], asks for an online algorithm to reconfigure an unordered linked list datastructure, such that a sequence of lookup requests is served optimally and at minimal reconfiguration costs (i.e., pointer rotations). It is well-known that a simple *move-to-front* strategy, which immediately promotes each accessed element to the front of the list, is *dynamically optimal*, that is, has a constant competitive ratio.

This paper initiates the study of pairwise communication problems in a dynamic network reconfiguration model. This model consists of a a set of communication nodes V, and a graph called the *host network*, denoted $H = (N, L)$ where $L \subseteq N \times N$. The communication nodes are 'hosted' on H, denoted with an injection $h : V \to N$ called a *configuration*. Without loss of generality, we assume that $|V| = |N|$, so that every configuration is a bijection. Two nodes $u, v \in V$ are *connected* on a configuration h if $(h(u), h(v)) \in L$.

Now, given a *communication sequence* $\sigma = \sigma_1, \sigma_2, ...,$ where $\sigma_i \in V \times V$, the pairwise communication problem asks to serve all communication requests in order. A communication request is served if the two constituent nodes are connected. Additionally, the network is allowed to be *reconfigured*, by migrating any two nodes which are connected. That is, given a configuration h in which u, v are connected, migrating them corresponds to producing a new configuration h' where $h'(u) = h(v)$, $h'(v) = h(u)$, and $h'(w) = h(w)$ for $w \neq u, v$. For simplicity, we assume in this paper that both serving a communication request and reconfiguring the network are constant cost operations. Note that in general migrating two nodes is likely (a large constant factor) more expensive than serving a single communication. However, with the assumption that communicating and migrating are the same up to a (large) constant factor, one can think of the communication cost between two arbitrary node as simply the number of reconfigurations necessary to make them (temporarily) connected. With this simple cost model, we can thus phrase the pairwise communication problem as:

Definition 1 (Pairwise Communication Problem). *Given a host network with an initial configuration, give an algorithm that serves all communication requests from a sequence σ that minimises the total reconfiguration cost.*

In the *offline* version of this problem, σ is given in advance, whereas in an *online* setting, a communication request σ_i is only revealed after σ_{i-1} has been served. A *competitive analysis* compares an online algorithm *ON* to an offline algorithm *OFF*. The ultimte goal is to devise online algorithms *ON* for the pairwise communication problem which minimise the competitive ratio ρ:

$$\rho = \max_{\sigma} \frac{\text{cost}(ON(\sigma))}{\text{cost}(OFF(\sigma))}$$

2 Results & Open Problems

In this paper, we study host networks with the topology of a d-dimensional grid. The primary problem we investigate is therefore:

Definition 2 (Distributed Grid Update). *What is the competitive ratio for for the pairwise communication problem for host networks with a d-dimensional grid topology?*

We show that for such networks there is a $\Omega(\log n)$ lower bound on the competitive ratio. Furthermore, in pursuit of a non-trivial upper bound, we show that for communication sequences with *linear demand*, there is an algorithm that is $O(\log n)$-competitive. A communication sequence is said to have linear demand if there is a configuration h that serves each request (without reconfigurations), or more formally:

Definition 3 (Linear Demand). *Let $R(\sigma)$ denote the set of communication requests interpreted as edges over V. A sequence σ has linear demand over a network $H = (N, L)$ if there exists a configuration h such that for all $(u, v) \in R(\sigma)$, it holds that $(h(u), h(v)) \in L$.*

From a lower bound perspective, our main result is that the competitive ratio is at least $\Omega(\log n)$. We show this by explicitly constructing a hard sequence:

Theorem 1. *For every online algorithm* ON *solving* Distributed Grid Update *on a grid of n nodes in total and for every $0 < \varepsilon \leq 1$, there is a sequence σ_{ON} of length $\mathcal{O}(\varepsilon n^{1+\varepsilon} \log n)$ such that cost$(\text{ON}(\sigma_{\text{ON}})) = \Omega(\varepsilon n^{1+1/d} \log n)$. The resulting request graph $R(\sigma_{\text{ON}})$ is a d-dimensional grid graph.*

Since $R(\sigma_{ON})$ is a grid graph, an offline algorithm can simply configure to the configuration h that serves all requests (i.e. σ_{ON} has linear demand). From an

arbitrary starting configuration, it takes $\Omega(n^{1+1/d|})$ to configure h, which dominates the cost for serving all requests as long as $\varepsilon \leq 1/d$. Therefore, we find the following lower bound on ρ:

$$\rho \geq \frac{\text{cost}(ON(\sigma_{ON}))}{\text{cost}(OFF(\sigma_{ON}))} = \Omega(\log n)$$

Because we rely on linear demand, our technique does not leverage the full power available to an offline algorithm: namely reconfigurations in between communication requests. As future work, we pose the following open problem:

Problem 1. Construct a sequence σ with nonlinear demand for Distributed Grid Update, such that any online algorithm is a factor $\omega(\log n)$ off the optimal offline algorithm.

From an upper bound perspective, we investigate Distributed Grid Update restricted to line topologies, dubbed Distributed List Update. We show that in the very restricted case of linear demand, there is a matching upper bound to the competitive ratio:

Theorem 2. *There is an algorithm* Gread *solving* Distributed List Update*, such that if σ has linear demand, then*

$$\textit{cost}(\textsc{Gread}(\sigma)) = \mathcal{O}(m + nk \log k)$$

where $m = |\sigma|$ and $k = |R(\sigma)|$

Again, it is the restriction to linear demand that allows for this tight upper bound. However, we see this as an initial step to finding any nontrivial upper bound for the general case:

Problem 2. Given a sequence σ for Distributed List Update with arbitrary request graph $R(\sigma)$, is there an algorithm with competitive ratio $o(n)$?

Note that a competitive ratio of $o(n)$ means *anything* better than a trivial algorithm. However, a better understanding of the behaviour of offline algorithms for Distributed Grid Update on nonlinear demand is required to answer this question.

References

1. Avin, C., Loukas, A., Pacut, M., Schmid, S.: Online balanced repartitioning. In: Proc. 30th International Symposium on Distributed Computing (DISC) (2016)
2. Avin, C., Schmid, S.: Toward demand-aware networking: A theory for self-adjusting networks. In: ACM SIGCOMM Computer Communication Review (CCR) (2018)
3. M. Ghobadi et al.: Projector: Agile reconfigurable data center interconnect. In: Proc. ACM SIGCOMM. pp. 216–229 (2016)
4. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Communications of the ACM **28**(2), 202–208 (1985)