On the Value of Coordination in Distributed Decision Making*

Sandy Irani[†] Yuval Rabani[‡]

Keywords: Analysis of Algorithms, Distributed Computation, Competitive Analysis, Load Balancing, Virtual Circuit Routing.

Abstract

We discuss settings where several "agents" combine efforts to solve problems. This is a well-known setting in distributed artificial intelligence. Our work addresses theoretical questions in this model which are motivated by the work of Deng and Papadimitriou [11]. We consider optimization problems, in particular load balancing and virtual circuit routing, in which the input is divided among the agents. An underlying directed graph, whose nodes are the agents, defines the constraints on the information each agent may have about the portion of the input held by other agents. The questions we discuss are: Given a bound on the maximum out-degree in this graph, which is the best graph? What is the quality of the solution obtained as a function of the maximum out-degree?

*An earlier version of this paper appeared in *Proceedings of the 34th IEEE Symposium on Foundations of Computer Science*, Palo Alto, 1993, under the title "On the value of information in coordination games."

†Supported in part by NSF Grant CCR-9309456. Address: Department of Information and Computer Science, University of California, Irvine, CA 92717. Email: `irani@ics.uci.edu` .

‡Address: Lab. for Computer Science, MIT, 545 Technology Square, Cambridge, MA 02139. Work done while at ICSI, Berkeley, and supported by a Rothschild post-doctoral fellowship. Email: `rabani@theory.lcs.mit.edu` .

1 Introduction

In recent years, there has been a great deal of research activity focused on analyzing algorithms which must compute using partial information about the problem to be solved. Much of this research effort has focused on *on-line* algorithms, where the limitation is due to temporal constraints: the input is arriving a piece at a time, and output must be produced before all the input arrives. The study of on-line algorithms is motivated by the fact that many problems which arise in a wide variety of settings are inherently on-line (i.e., one doesn't have the luxury of being able to collect all the information about an the instance of the problem before a partial answer must be produced). However, part of the reason for the recent interest in this area is the introduction of an appealing means of evaluating on-line algorithms called *competitive analysis* [18, 15]. The idea is to determine the quality of an on-line algorithm by comparing its performance to the performance of the optimal algorithm that can see the entire input in advance. Thus we measure what is lost by solving the problem on-line.

The work in this paper follows a model proposed by Deng and Papadimitriou [11] and further discussed by Papadimitriou and Yannakakis [17] who extend the use of competitive analysis to a more general setting than on-line algorithms. They discuss settings where several "agents" combine efforts to solve problems. The idea is that global information about a problem to be solved may be lacking due to spatial (or other) constraints. The framework suggested in [11] is to model specific constraints in the availability of information and study the solution quality that can be obtained under these information regimes. In [17], the information regime is determined by the input. Linear programming is considered, where each agent is responsible for a variable or a set of variables, and sees all constraints involving those variables.

We take a different approach that is suitable for the case that information is available at a price. Rather than analyzing particular constraint structures, we focus on the best constraint structure given a bound on the amount each agent communicates. We address the following questions. To what extent is it useful for the agents to communicate information about the piece of the input that each holds? What is the most effective pattern of communication? If some a priori information about the problem instance is known, how can this information be used to improve the solution quality?

1.1 The Model

We consider optimization problems in the following context: A set of agents $A = \{a_0, a_1, \ldots, a_{n-1}\}$ are given an instance I of an optimization problem. Each agent is presented with a portion of the input: a_i gets input I_i, where $I = \cup_i I_i$. We assume that the objective function is not part of the input and is known to all agents.

A *strategy* S for the agents is a pair (G, D), where G is a directed graph and D is a set of algorithms, one for each agent. We refer to G as the *knowledge graph* because it represents the information available to each agent: a directed edge (a, b) in G means that agent a knows of the input portion given to agent b. Each agent's decisions are a function only of the input it knows: a set of algorithms D is *valid* for a knowledge graph G if the algorithm for each agent a_j is a function only of I_j and all I_k such that $(j, k) \in E$. If $S = (G, D)$ is a strategy, then D must be valid for G. In some cases, we refer to an undirected knowledge graph in which case an edge between agents a and b indicates that a and b know each other's input.

We denote by $\text{cost}_S(I)$ the value of the objective function for a strategy S on input I, and we denote by $\text{cost}(I)$ the value of the objective function for the optimal, global strategy on input I. A strategy S is *c-competitive* if for every I, $\text{cost}_S(I) - c \cdot \text{cost}(I)$ is bounded by a constant. The *competitive ratio* of S denoted c_S is the infimum over all c such that S is c-competitive.

If the knowledge graph is predetermined, then the goal is to devise the best algorithm with the given limitation in information. Thus, for a given knowledge graph G, we would like to choose the best set of algorithms D that are valid for that graph. The competitive ratio for a knowledge graph denoted c_G is the infimum of $c_{(G,D)}$ over algorithms D that are valid for G. On the other hand, if complete information is available (that is, any two agents can communicate) but at a cost, we would like to determine to what extent the solution can be improved with more information. Thus, we consider strategies that are constrained by limiting the maximum degree of a vertex in G. If G has maximum degree r, we call S an *r-strategy*. c_r is the infimum of c_S over all r-strategies. What pattern of communication and what algorithm are best for a given limitation in communication: for a given r what r-strategy achieves c_r? To what extent is communication useful: how does c_r decrease as r increases? We study these general issues with respect to two specific problems: load balancing and virtual circuit routing.

Awerbuch *et al.* [3] study a problem very similar in flavor to the problems we consider here. They consider the number of steps necessary to broadcast a message in a fixed network. They show bounds on the number of steps necessary as a function of the radius of the network graph each vertex knows. Unlike our model, the processors do not get to choose the information they acquire. However, for the problems we consider, it is always optimal for the agents to communicate by grouping themselves into disjoint cliques and completely sharing information within a clique.

There are several applications to this model. We mention some of them here.

Parallel Programming. It is now clear that realistic models of parallel computation must address communication overhead as well as processing time [1, 10, 12, 16, 19]. It seems easier to design and implement a parallel program where the parallel tasks are oblivious to each other as much as possible. Coordination between parallel tasks requires additional programming and increases communication overhead. Our work speaks to the tradeoff between the amount of coordination in a parallel program and the effectiveness of that program in solving specific problems. Particularly, the design of system services such as batch execution might benefit from this analysis.

High speed network management. High speed networks are expected to serve a large number of users bidding for a variety of services. The allocation of network resources by a centralized network manager becomes impractical under such conditions. As pointed out by [17], the multiple agents model is suitable for discussing performance degradation due to the distributed nature of network management. Specifically, our results relate to the following questions: Network managers are requested to allocate virtual connecting paths between pairs of sites. Each virtual circuit consumes a fixed bandwidth. What is the required capacity of network links and switches to handle the expected traffic? How does this capacity change as a function of communication among network managers? What network structures support distributed management?

Large scale planning. The question of cooperation among communicating problem solvers is considered fundamental in distributed AI, as it addresses planning problems in a large scale system or organization that is faced with a rapidly changing environment. See [6] for a comprehensive collection of papers in the field. The AI approach tends to be either qualitative or experimental. Recent theoretical results [11, 17] as well as this work focus on quantitative analysis of such problems.

1.2 Outline of Results

We consider the load balancing problem discussed in [11]. Each agent gets a set of jobs to be executed, where the length of each job is known in advance. The agents redistribute the jobs among themselves. Their goal is to minimize the maximum load on an agent. The optimal strategy clearly divides the jobs evenly among all the agents, or as evenly as possible given the granularity of the job lengths.

Deng and Papadimitriou give a complete analysis of the problem of three agents scheduling jobs on two processors for all possible knowledge graphs. They also show that for an arbitrary number n of agents distributing jobs among themselves, when the agents do not communicate at all (i.e., the knowledge graph has no edges), then there is a way for each agent to redistribute its jobs that achieves a competitive ratio of $2\sqrt{n}$. Furthermore, for any deterministic strategy, there is a way of assigning n jobs of length 1 to the n agents such that some agent receives at least $\sqrt{n}$ jobs. Thus if G has no edges, then $\sqrt{n} \leq c_G \leq 2\sqrt{n}$.

We generalize their results to show that for a fixed knowledge graph G, $\sqrt{\alpha(G)} \leq c_G \leq 2\sqrt{\phi(G)}$, where $\alpha(G)$ is the size of the maximum independent set of G and $\phi(G)$ is the size of the minimum clique cover in G. Thus, by choosing G to be a collection of disjoint $(r+1)$-cliques, $\sqrt{n/(r+1)} \leq c_r \leq 2\sqrt{n/(r+1)}$. The factor of two can be reduced when all the job lengths are identical.

As one might suspect, randomization is a very powerful tool in this setting. Deng and Papadimitriou show for the empty knowledge graph that if each agent sends each job to a random destination, then the competitive ratio is $\log n/\log\log n$. We show an asymptotically matching lower bound (also shown independently by Alon [2]). Furthermore, we consider r-strategies for all r. We show tight bounds of $c_r \in \Theta(\log(\frac{n}{r})/\log\log(\frac{n}{r}))$. The lower bound holds for any distribution of r-regular graphs; i.e., even when global information available to all agents is hidden from the adversary. For example, the lower bound holds even if the agents can organize themselves into random collections of disjoint $(r+1)$-cliques. The upper bound follows, as in the deterministic case, from an algorithm that gives $c_G = O\left(\log\phi(G)/\log\log\phi(G)\right)$

for all knowledge graphs G. The upper bound requires that agents which share their pieces of input can also toss common coins (but no global coins are needed).

We also consider questions that relate to virtual circuit routing. The problem is to route permutations in a network where each agent is responsible for selecting the route of a single input. The goal is to minimize the node or edge congestion. When no information is available to the agents besides their own destination assignment, this is the well-studied question of *oblivious* routing. If the paths for each input-output pair are precomputed by some central algorithm, we have the problem of *global* routing – also, a very well-understood problem. We consider routing where the available information is in the spectrum between the oblivious and the global cases. Each agent (which represents a single input vertex in the network) knows its own destination and the destinations of some of the other agents and must decide on its path using the available information. The knowledge graph represents what information is available to which agents and we determine the benefit obtained when each agent has degree at most r in the knowledge graph.

We consider N-node, degree-d networks with n input and n output nodes. We assume that the network is optimal for the required task; i.e., the agents may choose the structure of the network. For that reason, we do not give a competitive analysis but a worst case analysis. We show an r-strategy for routing in a $\log n$-dimensional Beněs network [4, 20] with maximum edge congestion of $\sqrt{\frac{n}{r}}$. The lower bounds on oblivious routing of [7, 14] can be modified to show lower bounds of $\frac{1}{2d}\sqrt{\frac{n}{N}\frac{n}{r}}$ on edge congestion and $\frac{1}{2}\sqrt{\frac{n}{N}\frac{n}{(d+1)r}} - \frac{n}{2Nr}$ on node congestion. We show a lower bound on node congestion of $\Omega\left(\frac{\log(\frac{n}{r})}{\log\log(\frac{n}{r})}\right)$ for randomized r-strategies. The lower bound follows the lower bound on oblivious single-port routing of Borodin et al. [9]. All of our bounds match the previously known bounds for $r = 1$ (oblivious routing) and $r = n$ (global routing).

2 Load Balancing

Each agent decides deterministically where to send its jobs based on the set of jobs it has and the set of jobs each of its neighbors in the knowledge graph has. Let $\phi(G)$ be the size of the minimum clique cover for a graph G and $\alpha(G)$ the size of the maximum independent set of G. We prove the following theorem.

Theorem 1 *For every graph G, $\sqrt{\alpha(G)} \leq c_G \leq 2\sqrt{\phi(G)}$.*

Proof. Let S be an independent set. Let $|S| = s$. Give $n/\sqrt{s}$ jobs to each agent in S. After the jobs have been redistributed, some agent a will get $\sqrt{s}$ jobs. Pick a subset of at most $\sqrt{s}$ agents such that the total number of jobs given to agent a by agents in the subset is at least $\sqrt{s}$. Now instead, give $n/\sqrt{s}$ jobs only to agents in the subset. Since there are at most n jobs total, the optimal cost is one. Agent a still gets $\sqrt{s}$ jobs.

Let $\phi(G) = \Phi$. Partition the vertices into Φ cliques, $c_0, \ldots, c_{\Phi-1}$. A clique acts as one agent since all the agents know the inputs of the other agents in the clique. Fix an arbitrary ordering on the agents: $a_0, ..., a_{n-1}$. Let $m = n/\Phi$. Each clique c_i divides its tasks into at most n groups so as to minimize the maximum length of a group's tasks. Then it sorts the groups in decreasing order by length. Clique c_i sends the jobs in group k to agent $a_{\lfloor mi \rfloor + k \pmod{n}}$.

Fix some agent a_j. Let i_j be the number that satisfies $\lfloor mi_j \rfloor \leq j < \lfloor m(i_j+1) \rfloor$. Let t_i denote the number of jobs given to agent j from the clique $c_{i_j - i \pmod{\Phi}}$ (the clique that precedes clique i_j in the ordering by i). Fix some k such that $1 \leq k < \Phi$. We divide the work sent to agent a_j into two parts: $T_1 = \sum_{i=0}^{k-1} t_i$, and $T_2 = \sum_{i=k}^{\Phi-1} t_i$. The total job length sent to agent a_j is $T = T_1 + T_2 \leq 2\max\{T_1, T_2\}$. We can lower bound the cost to the optimal strategy by $\max_{0 \leq i < k} t_i$ because each clique divides its jobs into groups so that the maximum total length in any group is minimized. If $T_1 \geq T_2$,

$$\frac{T}{\max_{0 \leq i < k} t_i} \leq \frac{2T_1}{\max_{0 \leq i < k} t_i} \leq 2k.$$

Clique $c_{i_j - i \pmod{\Phi}}$ sends at least t_i jobs to agents $\lfloor (i_j - i)m \pmod{n} \rfloor, \ldots, \lfloor mi_j \rfloor$. Thus the total number of jobs originating at clique $c_{i_j - i \pmod{\Phi}}$ is at least $(\lfloor mi_j \rfloor - \lfloor m(i_j - i) \pmod{n} \rfloor + 1)t_i \geq mit_i$. The total length of all the jobs in the system is at least $\sum_{i=k}^{\Phi-1} mit_i \geq mk \sum_{i=k}^{\Phi-1} t_i$. Thus, the optimal solution sends jobs of length at least

$$\frac{mk \sum_{i=k}^{\Phi-1} t_i}{n} = \frac{\sum_{i=k}^{\Phi-1} t_i}{\Phi/k} = \frac{kT_2}{\Phi}$$

to some agent. Thus, if $T_2 \geq T_1$, $\frac{T}{kT_2/\Phi} \leq \frac{2T_2}{kT_2/\Phi} = 2\Phi/k$. Picking k to minimize the maximum of the two bounds $2\Phi/k$ and $2k$, we get $k = \sqrt{\Phi}$ which yields an upper bound of $2\sqrt{\Phi}$. ∎

Corollary 2 $\sqrt{\frac{n}{r+1}} \leq c_r \leq 2\sqrt{\frac{n}{r+1}}$.

Proof. The lower bound follows from the fact that any graph with maximum degree r has an independent set of size at least $n/(r+1)$ (the greedy algorithm finds such a set). For the upper bound, partition the set of agents into disjoint subsets of size $r+1$ each. The knowledge graph consists of $n/(r+1)$ complete graphs, one on each of the subsets. ∎

Remark. The upper bound can be improved to $\sqrt{2\Phi}$ when all the job lengths are the same. When the jobs lengths are not the same, it is an NP-hard problem to divide a set of jobs into at most n groups so as to minimize the maximum total length in any group. However, there is a polynomial time approximation algorithm which comes within a factor of 4/3 of the optimal solution [13]. Thus, the above upper bound can be achieved by polynomial time bounded agents with an extra factor of 4/3 in the ratio.

A natural question to ask is whether the upper or lower bound of Theorem 1 is tight. In order to answer this question, we need to examine a class of graphs whose maximum independent set and minimum clique cover differ. The distribution $\mathcal{G}(n, \frac{1}{2})$ is the distribution over all n node undirected graphs where each pair of nodes is adjacent independently with probability $\frac{1}{2}$. If G is drawn according to $\mathcal{G}(n, \frac{1}{2})$, then with high probability, $\phi(G) = \Omega(n/\log n)$ and $\alpha(G) = O(\log n)$ [5]. These facts combined with the following claim imply that the upper bound of Theorem 1 is not tight for the case when all job lengths are 1. We suspect that the lower bound is not tight either — that there are graphs G for which c_G lies strictly between $\sqrt{\alpha(G)}$ and $\sqrt{\phi(G)}$.

Claim 3 *If G is drawn at random from $\mathcal{G}(n, \frac{1}{2})$, then with high probability, $c_G = O(n^{\frac{1}{3}} \log n)$ when all job lengths are uniform.*

Proof. We can assume that no agent receives more than n jobs. If an agent receives x jobs, it can distribute $n\lfloor x/n \rfloor$ jobs evenly among the agents and it is left with the problem of distributing the remaining $x \bmod n$ jobs.

An *i-adversary* gives each agent either 0 jobs or x jobs where $2^i \leq x < 2^{i+1}$. Suppose that for every $0 \leq i \leq \log n$, we can achieve a ratio of c

against an i-adversary. Then we can achieve a ratio of $c(\log n+1)$ against any adversary. Let the *i-agents* be those agents that receive y jobs such that $2^i \leq y < 2^{i+1}$. Each i-agent follows the strategy against the i-adversary with the following change. For every non-i-agent whose input it can "see," it assumes that that agent received 0 jobs. Let M_i be the number of jobs given to the agent who receives the maximum number of jobs from i-agents after the jobs have been redistributed. Let X_i be the total number of jobs given to i-agents by the adversary. Let $t=\log n$. The agent with the maximum number of jobs after redistribution has no more than $\sum_{i=0}^{t} M_i$ jobs. The optimal solution gives at least $\lceil(\sum_{i=0}^{t} X_i)/n\rceil$ jobs to some agent. We are guaranteed that $M_i/\lceil X_i/n\rceil \leq c$.

Since $\sum_{i=0}^{t}\lceil(X_i/n)\rceil \leq (t+1)\lceil(\sum_{i=0}^{t} X_i)/n\rceil$, we have that

$$\begin{aligned}\frac{\sum_{i=0}^{t} M_i}{\lceil\sum_{i=0}^{t}\frac{X_i}{n}\rceil} &\leq \frac{(t+1)\sum_{i=0}^{t} M_i}{\sum_{i=0}^{t}\lceil\frac{X_i}{n}\rceil}\\ &\leq (t+1)\cdot\max_i\left\{\frac{M_i}{\lceil\frac{X_i}{n}\rceil}\right\} \leq c\cdot(\log n+1)\end{aligned}$$

Now for every $0 \leq i \leq \log n$, we show a strategy against an i-adversary. Then we show that with high probability, G has a certain property which ensures that each strategy achieves a ratio of $O(n^{\frac{1}{3}})$. Say that G has property A if for every subset of $8n^{1/3}$ nodes in G, the subgraph induced by that subset has more than $4n^{2/3}$ edges. We have the following lemma:

Lemma 4 *When G is chosen according to $\mathcal{G}(n,\frac{1}{2})$, then*

$$\Pr[G \textit{ does not have property } A] \leq 2^{n^{1/3}\left(8\log n-5n^{1/3}/3\right)}.$$

Proof. Let $s=n^{1/3}$. Consider a fixed subset S of $8s$ vertices. What is the probability that there are at most $4s^2$ edges in the subgraph induced by S? There are at least $\binom{8s}{2} \geq 28s^2$ edges slots. Thus the expected number of edges is at least $14s^2$. Using Chernoff bounds, the probability that a fixed subset of $8s$ vertices induces a subgraph with no more than $4s^2$ edges is at most

$$e^{-(10s^2)^2/2\cdot 28s^2} \leq 2^{-5s^2/3}.$$

Thus, the probability that there is a subset of size $8s$ vertices in the graph whose induced subgraph has fewer than $4s^2$ edges is at most

$$\binom{n}{8s} 2^{-5s^2/3} \le n^{8n^{1/3}} 2^{-5n^{2/3}/3} \le 2^{n^{1/3}\left(8\log n - 5n^{1/3}/3\right)}. \qquad \blacksquare$$

Now we will show the strategy against an i-adversary. Our strategy gives the desired ratio if G has property A. Therefore, if property A holds, the strategies for all i give the desired ratio. Let $m = 2^{\lceil \log n \rceil}$. There will be n agents sending jobs and and m *receivers*. The jobs sent to receiver j are sent to agent $j \pmod n$. An agent gets no more than twice the load of the heaviest loaded receiver. We will determine the ratio of the most heavily loaded receiver in the distributed solution to the most heavily loaded agent in the optimal solution. Let $X = 2^i$. Let x_j be the number of jobs that originate with agent j. Since we are playing against an i-adversary, $x_j = 0$ or $X \le x_j < 2X$ for all j.

If $X \ge n^{2/3}$, then agent j sends a job to receiver $k + j \bmod n$ for all $1 \le k \le x_j$. If some agent gets a jobs after redistribution, then there are at least aX jobs among all the agents. The optimal solution gives at least aX/n to some agent. This gives a ratio of at most n/X.

If $X \le n^{1/3}$, then each agent keeps all the jobs that it receives.

If $n^{1/3} \le X \le n^{2/3}$, then divide the receivers into m/X groups of X consecutive receivers. The agents are also divided into consecutive groups of X. There are only $\lceil n/X \rceil$ such groups and the last group may have fewer than X agents.

There are two cases:

Case 1. If an agent in group i gets jobs and is adjacent to no more than $n^{1/3}$ agents with jobs in its group, then it distributes its jobs evenly among the receivers in group i. In this case, an agent gives at most two jobs to each receiver.

Case 2. If an agent j is adjacent to at least $n^{1/3}$ agents with jobs in its group, it sends its jobs to receiver $(Xk + (j \bmod X)) \pmod m$ for $0 \le k < x_j$. In other words, each agent can be specified by the group to which it belongs (j div X) and its number within the group ($j \pmod X$)). Starting with group 0 and cycling through the m/X groups of receivers, agent j gives its jobs to the receiver in each group which has the same number within its group.

Now consider a group of X agents with more than $8n^{1/3}$ agents that fall into case 1. Pick $8n^{1/3}$ of them and call this set S. Every agent in S has fewer that $n^{\frac{1}{3}}$ edges to agents in S. That means that the subgraph induced by S has no more than $4n^{2/3}$ edges (since the sum of degrees is twice the number of edges); i.e., G does not have property A. Thus, by Lemma 4, with high probability G has the property that it is impossible to have more than $8n^{1/3}$ agents that fall into case 1. For the remainder of the proof, we assume that this is the case. Using the fact that an i-agent from Case 1 gives at most two jobs to each agent in its group, we can conclude that the number of jobs given to an agent by i-agents from Case 1 is at most than $16n^{1/3}$.

So suppose that some receiver j gets l jobs from Case 2 agents. Receiver j only gets these jobs from agents whose number is congruent to j mod X. Each of these agents is in a different group and spreads its jobs as evenly as possible among the groups of receivers. Since each agents starts with at most $2X$ jobs and there are m/X groups, at most $\lceil 2X^2/m \rceil \leq \max\{1, 4X^2/m\}$ jobs that end up with agent j come from the same group. Thus, there are at least $\min\{l, lm/4X^2\}$ groups where $n^{1/3}$ of the agents get jobs. Each such group has at least $Xn^{1/3}$ jobs. Thus, the total number of jobs is at least $Xn^{1/3}\min\{l, lm/4X^2\}$, and the optimal solution gives at least $(X/n^{2/3})\min\{l, lm/4X^2\}$ to some agent. Thus, the competitive ratio due to Case 2 is at most $\max\{n^{2/3}/X, 4X/n^{1/3}\}$. For $n^{1/3} \leq X \leq n^{2/3}$, the ratio is at most $4n^{1/3}$. ■

We show the following upper bound on randomized strategies.

Theorem 5 *For every knowledge graph G there is a randomized load balancing strategy whose competitive ratio is in $O\left(\frac{\log(\Phi)}{\log\log(\Phi)}\right)$ where $\Phi = \phi(G)$.*

Proof. Let $m = \lceil n/\Phi \rceil$. Pick a clique cover of G of size Φ. We assume that the agents within a clique know the set of jobs assigned to the other agents in the clique as well as their random bits. Thus, each clique operates as a single agent. The agents will be divided into Φ sets of size m. Each agent will be indexed by a pair (i, j) where $0 \leq i \leq \Phi - 1$ and $0 \leq j \leq m-1$. i indicates the set to which the agent belongs and j indicates the place within the set.

Each clique c_l divides its tasks into at most n sets such that the maximum length of a set's tasks is minimized. Then it sorts the sets in decreasing order of length. Each set will be indexed by a triplet (i, j, l) where $0 \leq i, l \leq \Phi - 1$

and $0 \leq j \leq m-1$. The third index denotes the clique where the jobs originate. The first two indices identify the specific set at clique l. Let $s(i,j,l)$ be the total length of the jobs in set (i,j,l). They are sorted so that $s(i,j,l) \geq s(i',j',l)$ if $i < i'$ or if $i = i'$ and $j \leq j'$. For each $i \in \{0,\ldots,\Phi-1\}$, clique c_l draws $k \in \{0,\ldots,\Phi-1\}$ uniformly at random. Then for all $j \in \{0,\ldots,m-1\}$, it sends all the jobs in set (i,j,l) to agent (k,j).

Denote the optimal cost for this instance by OPT. Denote the cost due to the distributed algorithm by the random variable D. We wish to show that $E[D] \in O((\log\Phi/\log\log\Phi)OPT)$.

Consider a related problem: we have Φ agents. Agent l gets a set of Φ jobs of lengths $s(0,0,l)$, $s(1,0,l)$, …, $s(\Phi-1,0,l)$. (Some of these could be of length 0). Let OPT' be the cost when the jobs are optimally distributed among the Φ agents. Let D' be the random variable denoting the cost of the distributed algorithm which sends each job to a random agent. The result of [11] gives that $E[D'] \in O((\log\Phi/\log\log\Phi)OPT')$. Clearly, D and D' are identically distributed, so $E[D] = E[D']$. We show that $OPT \geq \frac{1}{3}OPT'$, which completes the proof.

Divide the jobs from the second problem into two sets:

$$T_1 = \sum_{l=0}^{\Phi-1} s(0,0,l)$$

$$T_2 = \sum_{i=1}^{\Phi-1}\sum_{l=0}^{\Phi-1} s(i,0,l)$$

Let $MAX = \max_{0\leq l\leq\Phi-1} s(0,0,l)$ Clearly, $MAX \geq T_1/\Phi$. We claim two facts:

1. $OPT \geq \max\{MAX, \frac{T_2}{\Phi}\}$;

2. $OPT' \leq MAX + \frac{T_1+T_2}{\Phi}$.

The theorem follows from the two claims because

$$OPT' \leq 2MAX + \frac{T_2}{\Phi} \leq 3OPT.$$

To prove the first claim, observe that in the first problem each clique divides the jobs into sets so as to minimize the maximum length of the jobs in any set. So the length of the jobs in any set is a lower bound on the optimal solution. The second part of the 'max' in claim 1 follows from

$$OPT \geq \frac{1}{n} \sum_{j=0}^{m-1} \sum_{i=0}^{\Phi-1} \sum_{l=0}^{\Phi-1} s(i,j,l)$$

Since for all $1 \leq i < m$, $\sum_{j=0}^{m-1} s(i,j,l) \geq m \cdot s(i+1,0,l)$,

$$\begin{aligned} &\geq \frac{m}{n} \sum_{i=1}^{\Phi-1} \sum_{l=0}^{\Phi-1} s(i,0,l) \\ &\geq \frac{mT_2}{n} = \frac{T_2}{\Phi}. \end{aligned}$$

To see the second claim, observe the discrepancy *disc* in the optimal solution. The discrepancy is the difference between the maximum load on an agent and the minimum load on an agent. Clearly, $disc \leq MAX$. Since $(T_1 + T_2 + \Phi \cdot disc)/\Phi$ is an upper bound on the optimal cost, the claim follows. ■

Corollary 6 *There are randomized load balancing r-strategies for all n, r with a competitive ratio in* $O\left(\frac{\log(\frac{n}{r})}{\log\log(\frac{n}{r})}\right)$.

Proof. Partition the set of agents into disjoint subsets of size $r+1$ each. The knowledge graph consists of $\lceil n/(r+1) \rceil$ complete graphs, one on each of the subsets. ■

Corollary 6 is tight up to a constant factor, as the following theorem shows.

Theorem 7 *If* $\sqrt[4]{\frac{r(n)}{n}} \longrightarrow 0$ *as* $n \to \infty$*, then for every sufficiently large n and for* $r = r(n)$*, the competitive ratio of every randomized load balancing r-strategy on n agents is in* $\Omega\left(\frac{\log(\frac{n}{r})}{\log\log(\frac{n}{r})}\right)$.

Remark. If $\sqrt[4]{\frac{r(n)}{n}} \geq \epsilon > 0$ for all n, then $\frac{n}{r(n)} \leq \epsilon^{-4}$. So, we get a lower bound of $\Omega\left(\frac{\log(\frac{n}{r})}{\log\log(\frac{n}{r})}\right)$ for all $r < n$. Alon [2] has independently shown a lower bound of $\Omega(\log n / \log\log n)$ for the special case of an empty knowledge graph. This bound follows from our proof as well.

Proof of Theorem 7. We need to consider only r-regular knowledge graphs (i.e., with *out-degree* r). Applying von Neumann's minimax principle (see [21, 8]), we show a probability distribution over inputs which beats every deterministic algorithm.

For the sake of completeness, we state and prove the exact claim that is needed (see [8, Lemma 7.2] for the original version).

Lemma 8 *Let $\tilde{I}$ be a probability distribution over a finite sample space of inputs, such that for every deterministic r-strategy S, $E_{\tilde{I}}[\mathrm{cost}_S(\tilde{I}) - c \cdot \mathrm{cost}(\tilde{I})] \geq 0$. Then, for every randomized r-strategy $\tilde{S}$ there exists an input instance $I = I(\tilde{S})$ for which*

$$\frac{E_{\tilde{S}}[\mathrm{cost}_{\tilde{S}}(I)]}{\mathrm{cost}(I)} \geq c.$$

Proof. Fix $\tilde{S}$. $\tilde{S}$ is a probability distribution over deterministic strategies S_χ. Since for every χ, $E_{\tilde{I}}[\mathrm{cost}_{S_\chi}(\tilde{I}) - c \cdot \mathrm{cost}(\tilde{I})] \geq 0$, we have that $E_\chi[E_{\tilde{I}}[\mathrm{cost}_{S_\chi}(\tilde{I}) - c \cdot \mathrm{cost}(\tilde{I})]] \geq 0$. We may switch the order of integration since all expectations are finite. We get $E_{\tilde{I}}[E_\chi[\mathrm{cost}_{S_\chi}(\tilde{I}) - c \cdot \mathrm{cost}(\tilde{I})]] \geq 0$, or $E_{\tilde{I}}[\mathrm{cost}_{\tilde{S}}(\tilde{I}) - c \cdot \mathrm{cost}(\tilde{I})] \geq 0$. Therefore, there exists $I \in \tilde{I}$ for which $\mathrm{cost}_{\tilde{S}}(I) - c \cdot \mathrm{cost}(I) \geq 0$. ■

The distribution we choose gives every agent d jobs independently with probability p and 0 jobs otherwise. We use $d = \sqrt[4]{r^3 n}$ and $\frac{4}{d} < p < \frac{4e}{d}$, so that $p(1-p)^r d = 4$. Notice that $d \in \omega(r) \cap o(n)$. The expected cost of the optimal algorithm is at most $8e$. Our goal is to show that the expected cost of any deterministic r-strategy is in $\Omega(\beta)$, where $\beta = \log(n/r)/\log\log(n/r)$.

Let $k = \log(\frac{d}{r})/\log\log(\frac{d}{r})$. By our choice of d, $k \geq \beta/4$. We need the following lemma.

Lemma 9 *If $r(n) \in o(d(n))$, and $Z = Z(n)$ is distributed according to the binomial distribution $B\left(\frac{4(r+1)}{d}, \frac{d}{4(r+1)}\right)$, then for n large enough,*

$$Prob[Z \geq k] \geq \frac{r}{d}.$$

Proof. Let $\lambda = E[Z] = 1$. $Prob[Z = k]$ (which is a lower bound for $Prob[Z \geq k]$) can be estimated using the Poisson approximation to the binomial distribution. It gives

$$Prob[Z = k] > p(k;\lambda) e^{-\frac{k^2}{d/4(r+1)-k} - \frac{\lambda^2}{d/4(r+1)-\lambda}},$$

where $p(k;\lambda) = e^{-\lambda}\lambda^k/k!$.

We can estimate $p(k;\lambda)$ for large k using Stirling's formula as follows.

$$p(k,1) = \frac{1}{e(k)!} \geq \frac{a}{\sqrt{k}}\left(\frac{e}{k}\right)^k,$$

for some constant a.

Also, $e^{-\frac{k^2}{d/4(r+1)-k}-\frac{\lambda^2}{d/4(r+1)-\lambda}} \longrightarrow 1$ as $n \to \infty$, so for sufficiently large n this is lower bounded by $\frac{1}{2}$.

We have, for sufficiently large n,

$$\log\left(\frac{2\sqrt{k}}{a}\left(\frac{k}{e}\right)^k\right) \leq k\log k \leq \log\left(\frac{d}{r}\right). \qquad \blacksquare$$

The condition that $\sqrt[4]{\frac{r(n)}{n}} \longrightarrow 0$ as $n \to \infty$, implies that $r(n) \in o(d(n))$.

We use Lemma 9 to bound a similar distribution. A vertex of G is said to be *chosen* if it gets jobs. A vertex is *isolated* if it is chosen and none of its neighbors are chosen.

Lemma 10 *Let A be a subset of the vertices of G. To each vertex $v \in A$, assign an integer weight W_v between 1 and $k/8$ such that $\sum_{v\in A} W_v \geq \frac{d}{4}$. Let Y be the sum of the weights of the isolated vertices in A. Let Z be as in Lemma 9. Then for n sufficiently large,*

$$Prob[Y \geq \frac{k}{2}] \geq \frac{1}{e^9}Prob[Z \geq k].$$

Proof. Recall that $k = \log(d/r)/\log\log(d/r)$. Let $s = d/4$. We may assume that $\sum_{v\in A} W_v = s$, otherwise we reduce weights and remove 0-weighted vertices from A until equality holds. For each vertex $v \in A$, we introduce W_v indicator variables. Each variable is set to 1 if its corresponding vertex is isolated and to 0 otherwise. Denote the indicator variables by $Y_1, \ldots, Y_s$. $Y = \sum Y_j$. For X a k-subset of $\{1,\ldots,s\}$, $V_Y(X)$ is the subset of vertices associated with $\{Y_i \mid i \in X\}$.

The proof proceeds in two steps. First, we relate the probability of the desired event in G to the probability of a similar event in a graph G_U, derived from G, where vertex weights are 1. Then we relate the probability of the desired event in G_U to the distribution of Z.

We form a graph G_U which is identical to G except that for every vertex $v \in A$, we have a set S_v of W_v nodes in G_U. If v and u are adjacent in G, then in G_U every vertex in S_u is adjacent to every vertex in S_v. Note that the degree in G_U is at most $kr/8$. Let $A_U = \cup_{v \in A} S_v$. Notice that $|A_U| = s$. Let U_i denote the indicator variable that is 1 if vertex i in A_U is isolated and 0 otherwise when each vertex of G_U is chosen independently of the others with probability p. Let $U = \sum U_j$.

Now consider a particular graph G_Z of s vertices consisting of disjoint $(r+1)$-cliques. Let $Z_1, \ldots, Z_s$ be the indicator variables which indicate for each vertex whether it is isolated when each vertex is chosen with probability p. Clearly, the distributions of Z from Lemma 9 and $\sum Z_j$ are identical.

We wish to show that

1. $Prob[Y \geq \frac{k}{2}] \geq Prob[U \geq k]/e^4$.
2. $Prob[U \geq k] \geq Prob[Z \geq k]/e^5$.

Proof of *Part (2):* Define

$$S_U = E\left[\sum_{X \subset \{1,\ldots,s\}, |X|=k} \prod_{j \in X} U_j\right].$$

Similarly

$$S_Z = E\left[\sum_{X \subset \{1,\ldots,s\}, |X|=k} \prod_{j \in X} Z_j\right].$$

Clearly, S_U is an upper bound on $Prob[U \geq k]$ and S_Z is an upper bound on $Prob[Z \geq k]$. First we prove that $S_U \geq S_Z/e^2$.

Examine a fixed k-set X of $\{1, \ldots s\}$. When does $\prod_{j \in X} U_j = 1$? This happens when X forms an independent set, all the vertices in X are chosen and all the vertices in the neighborhood set of X are not chosen. Notice that $|X| = k$ and the neighborhood set of X has at most $rk^2/8$ vertices. Thus if X forms an independent set, then $\prod_{j \in X} U_j = 1$ with probability at least $p^k(1-p)^{rk^2} \geq p^k/e \geq p^k(1-p)^{rk}/e$. (The first inequality follows from the fact that for sufficiently large n, $rk^2 \ll 1/p$. Since $(1-1/x)^{x-1} > e^{-1}$ for all $x \geq 2$, the inequality follows.)

Now let's look at the Z_js. If X forms an independent set, then the neighborhood set of X has size rk (because the graph is composed of disjoint

$(r+1)$-cliques). Thus if X is an independent set, then $\prod_{j\in X} Z_j = 1$ with probability exactly $p^k(1-p)^{rk}$.

Thus we have to prove that the number of k-sets X that are independent sets in G_U is at least $1/e$ times the number of k-sets X that are independent sets in G_Z. We do this by picking X at random and showing that $Prob[X \text{ is independent in } G_U] \geq e^{-1}$. We pick a random k-set, vertex by vertex in G_U. Let v_i denote the i^{th} vertex that is picked and V_i denote $\{v_1, \ldots, v_i\}$. The neighborhood set of V_i is denoted $N(V_i)$.

Thus, we have:

$$\begin{aligned}
& Prob[V_k \text{ is independent}] \\
= \; & \prod_{j=1}^{k} Prob[v_j \notin N(V_{j-1}) \mid V_{j-1} \text{ is independent }] \\
\geq \; & \prod_{j=1}^{k} \left(1 - \frac{rk(j-1)}{8(s-j+1)}\right) \geq \prod_{j=1}^{k} \left(1 - \frac{2rk(j-1)}{8s}\right) \\
\geq \; & \left(1 - \frac{rk^2}{4s}\right)^k \geq e^{-1}.
\end{aligned}$$

The last inequality holds for sufficiently large n, since $k \ll d/rk^2$. Therefore, we conclude that $S_U \geq S_Z/e^2$.

We now complete the proof that $Prob[U \geq k] \geq Prob[Z \geq k]/e^5$. Observe that

$$Prob[U \geq k] \geq S_U(1-p)^{s-k} \geq S_U/e^e \geq S_U/e^3.$$

The first inequality comes from the fact that the probability that there are exactly k isolated vertices in A is at least $S_U(1-p)^{s-k}$ (The lower bound is obtained by summing over all independent k-sets in G_U, the probability that the k-set is isolated and all other vertices in A_U are not chosen). The second inequality uses the fact that $p < e/s$ and that for sufficiently large n, $k > e$, and thus $s - k < (s/e - 1)e$. Therefore to complete the proof of *Part (2)*,

$$\begin{aligned}
Prob[U \geq k] \;&\geq\; \frac{1}{e^3} S_U \geq \frac{1}{e^5} S_Z \\
&\geq\; \frac{1}{e^5} Prob[Z \geq k].
\end{aligned}$$

Proof of *Part (1):* Let $\mathcal{I}$ be the set of independent sets in A of size at most $\frac{k}{2}$ whose weights sum to at least $\frac{k}{2}$. We want to relate the number of independent k-sets in A_U to the number of subsets in $\mathcal{I}$. To do that, we map every independent k-set X in A_U, to an independent set in $\mathcal{I}$ as follows: if $|V_Y(X)| \geq \frac{k}{2}$, then map X to an arbitrary size $\frac{k}{2}$ subset of $V_Y(X)$. Since the weight of a vertex is at least 1, the weight of the subset is at least $\frac{k}{2}$. If $|V_Y(X)| < \frac{k}{2}$, then map X to $V_Y(X)$. For any independent set $I \in \mathcal{I}$, there are at most $(\frac{k}{8})^{\frac{k}{2}}\binom{s}{k/2}$ independent sets mapped to I: there are at most $(\frac{k}{8})^{\frac{k}{2}}$ ways to pick the first $\frac{k}{2}$ vertices from the sets in G_U associated with the vertices of I and $\binom{s}{k/2}$ ways to pick the remaining vertices from A_U. Define

$$S_Y = \sum_{I \in \mathcal{I}} p^{|I|}(1-p)^{|N(I)|}.$$

We can lower bound S_Y by S_U/e as follows:

$$\begin{aligned}
S_U &\leq \sum_{X \text{ indep.}, |X|=k} p^k \\
&\leq \left(\frac{k}{8}\right)^{k/2} \binom{s}{k/2} \sum_{I \in \mathcal{I}} p^k \\
&\leq \left(\frac{k}{8}\right)^{k/2} \binom{s}{k/2} p^{k/2} \sum_{I \in \mathcal{I}} p^{k/2} \\
&\leq \left(\frac{k}{8}\right)^{k/2} \binom{s}{k/2} p^{k/2} \sum_{I \in \mathcal{I}} e p^{k/2}(1-p)^{kr/2} \\
&\leq \left(\frac{k}{8}\right)^{k/2} \binom{s}{k/2} p^{k/2} e S_Y.
\end{aligned}$$

We show that

$$\left(\frac{k}{8}\right)^{\frac{k}{2}} \binom{s}{k/2} p^{\frac{k}{2}} \leq 1,$$

thus concluding that $S_Y \geq S_U/e$. Observe that p was chosen so that $p(1-p)^r = \frac{4}{d} = \frac{1}{s}$. So $p^{\frac{k}{2}} = (\frac{1}{s})^{\frac{k}{2}}(1-p)^{-kr/2} \leq e(\frac{1}{s})^{\frac{k}{2}}$ for sufficiently large n.

$$\left(\frac{k}{8}\right)^{\frac{k}{2}} \binom{s}{k/2} p^{\frac{k}{2}}$$

$$
\begin{aligned}
&\leq \left(\frac{k}{8}\right)^{\frac{k}{2}} \frac{s^{\frac{k}{2}}}{(\frac{k}{2})!} e s^{-\frac{k}{2}} \\
&\leq \left(\frac{k}{8}\right)^{\frac{k}{2}} \left(\frac{2e}{k}\right)^{\frac{k}{2}} e \leq 1,
\end{aligned}
$$

where the inequalities hold for sufficiently large n. In a similar manner to the proof for Part (2),

$$
\begin{aligned}
Prob[Y \geq k/2] &\geq \frac{1}{e^3} S_Y \geq \frac{1}{e^4} S_U \\
&\geq \frac{1}{e^4} Prob[U \geq k]. \qquad \blacksquare
\end{aligned}
$$

This completes the proof of Lemma 10. We now proceed with the proof of Theorem7.

Consider the following n by n bipartite graph H. There is an edge between a vertex x on the left and a vertex y on the right for every job that agent x sends to agent y when agent x is isolated. There is an edge from vertex x on the left to vertex x on the right for every job that x keeps when x is isolated. Note that the degree of a left node in H is exactly d. If a vertex is isolated, then it sends its jobs according to the edges in H. We examine the load due to isolated vertices only.

Our proof proceeds in three steps. First, we address two special cases: (i) there is a right vertex in H of degree at least βd; (ii) there are at least d left vertices in H that have an edge of multiplicity $\beta/32$ or more. Then, we prove the theorem for graphs H that do not fall into either of these categories.

Lemma 11 *If H has a right vertex v of degree at least βd, then the expected number of jobs sent to v is at least 4β.*

Proof. Let $u_1, u_2, \ldots, u_k$ be the neighbors of v on the left. For $i = 1, 2, \ldots, k$, let W_i denote the multiplicity of the edge between u_i and v. $\sum W_i \geq \beta d$. Each of the u_i's is isolated with probability $p(1-p)^r$. Therefore, the expected number of jobs that v get is $p(1-p)^r \sum W_i \geq p(1-p)^r \beta d \geq 4\beta$.
$\blacksquare$

Lemma 12 *If there are at least d left vertices in H with an edge of multiplicity $\beta/32$ or more, then the expected maximum number of jobs an agent gets is at least $\beta/8e^3$.*

Proof. There is a set S of at least d vertices such that if any of them is isolated, some agent has a load of at least $\beta/32$. For $v \in S$, let E_v denote the event that v is isolated and the remaining vertices in S are not chosen. $Prob[E_v] \geq p(1-p)^{rk}(1-p)^{d/4e-1} \geq \frac{4}{de^3}$. (Recall that $p \leq 4e/d$ and for n sufficiently large $rk \ll 1/p$.) The probability that some vertex in S is isolated is at least $\sum_{v \in S} Prob[E_v] \geq 4/e^3$. Thus, the expected maximum load is at least $\beta/8e^3$. ∎

We now consider the remaining case, where all right vertices in H have degree at most βd, and at most d left vertices in H have edges of multiplicity $\beta/32$ or more. Remove all the vertices on the left which have an edge of multiplicity at least $\beta/32$. There are at least $n-d$ left vertices remaining.

We need the following lemma.

Lemma 13 *If the maximum degree of a right vertex in H is βd, then there exists a subgraph H' of H such that*

1. *The number of right vertices in H' is at least $\frac{d}{r}$.*
2. *The degree of each right vertex in H' is at least $\frac{d}{4}$.*
3. *For every two distinct right vertices x, y in H', their neighborhood sets do not intersect.*

Proof. Let $c = 2\beta d^3/r(n-2d)$. Define a graph I whose vertices are a subset of the right vertices of H as follows. Remove all right vertices of degree less than $d/2$. Remove edges till each remaining right vertex has degree exactly $d/2$. Call the remaining graph H''. Connect two right vertices by an edge iff the size of the intersection of their neighborhood sets in H'' is at least c. The degree of any vertex in I is at most d^2/c. The number of right vertices in H that have degree at least $d/2$ is at least $(n-2d)/2\beta$, because the maximum degree is βd and the sum of degrees is $(n-d)d$. Therefore, there is an independent set of size at least $(n-2d)c/2\beta d^2 \geq d/r$ in I. Take a subset of size d/r of the independent set together with the neighborhood sets of these vertices in H'' and the connecting edges. What we get is a collection of d/r stars. Each right vertex is a root of one star and has degree $d/2$. Remove from this collection any left vertex that participates in more than one star. Since each right vertex removes from each other star at most c left vertices, the resulting stars are vertex disjoint and have minimum root degree of at least $d/2 - cd/r$ which is at least $d/4$ for sufficiently large n. ∎

The neighborhood sets of the right vertices of H' induce a collection of d/r disjoint sets of vertices in G. We call this collection of sets $\mathcal{A} = A_1, A_2, \ldots, A_{d/r}$. To each vertex v that belongs to such a set, we assign it a weight W_v which is equal to the multiplicity of its edge to its adjacent right vertex in H'. Since we have removed the vertices incident to edges of high multiplicity, the weight of any vertex is at most $\beta/32 \leq k/8$. Each set has a total weight of at least $d/4$. Let a_i be the sum of the weights of the isolated vertices in set A_i. We wish to show that the expected maximum over all a_i is in $\Omega(\beta)$.

For the remainder of the proof, "adjacency" refers to adjacency in the knowledge graph G. $N(X)$ denotes the neighborhood set of a set of vertices X. A subset X of vertices is isolated (not isolated/chosen/not chosen) if every vertex in the set is isolated (not isolated/chosen/not chosen). Let E_j denote the event $(a_1 < k/2) \wedge (a_2 < k/2) \wedge \ldots \wedge (a_j < k/2)$.

Lemma 14 *Let $1 \leq j \leq d/r$. If $Prob[E_{j-1}] \geq 1 - \frac{1}{2e}$ then*

$$Prob[a_j \geq k/2 \mid E_{j-1}] \geq \frac{1}{2e^4} Prob[a_j \geq k/2].$$

Proof. Let $A = A_1 \cup A_2 \cup \cdots \cup A_{j-1}$. Let $E = E_{j-1}$. For each vertex $v \in A_j$ we introduce W_v indicator variables as in the proof of Lemma 10. Let $Y_1, \ldots, Y_s$ be those variables. S_Y is defined as in Lemma 10. Using the arguments from the proof of Lemma 10, we have that $Prob[a_j \geq k/2 \mid E] \geq E[S_Y \mid E]/e^3$. Also, $E[S_Y]$ is a trivial upper bound on $Prob[a_j \geq k/2]$. Therefore, it is sufficient to prove that $E[S_Y \mid E] \geq E[S_Y]/2e$. Fix an arbitrary independent set $I \in \mathcal{I}$. We need to prove that $Prob[I \text{ is isolated } \mid E] \geq Prob[I \text{ is isolated}]/2e$.

The event that I is isolated happens if and only if I is chosen and $N(I)$ is not chosen. Therefore, $p^{|I|} \geq Prob[I \text{ is isolated}]$. Also,

$$\begin{aligned} & Prob[I \text{ is isolated } \mid E] \\ = \; & p^{|I|} Prob[N(I) \text{ is not chosen } \mid E]. \end{aligned}$$

Since

$$\begin{aligned} Prob[N(I) \text{ is not chosen}] &= (1-p)^{|N(I)|} \\ &\geq (1-p)^{kr} \geq e^{-1}, \end{aligned}$$

we have that

$$\begin{aligned} & Prob[N(I) \text{ is not chosen } \mid E] \\ \geq\ & Prob[N(I) \text{ is not chosen } \wedge E] \\ \geq\ & Prob[N(I) \text{ is not chosen}] + Prob[E] - 1 \\ \geq\ & \frac{1}{2e}. \end{aligned}$$

Putting these facts together, we get that

$$Prob[I \text{ is isolated } \mid E] \geq \frac{p^{|I|}}{2e} \geq \frac{Prob[I \text{ is isolated}]}{2e}. \qquad \blacksquare$$

We can now complete the proof of the theorem. The condition of Lemma 10 holds for sufficiently large n. If there exists j, $1 \leq j \leq d/r$, for which $Prob[E_j] < 1 - 1/2e$, then the expected maximum load is at least $k/4e \geq \beta/16e$. Otherwise, we may use Lemma 14 to get

$$\begin{aligned} & Prob[\max a_j \geq k/2] \\ =\ & 1 - Prob[\forall j, a_j < k/2] \\ =\ & 1 - \prod_j Prob[a_j < k/2 \mid E_{j-1}] \\ =\ & 1 - \prod_j (1 - Prob[a_j \geq k/2 \mid E_{j-1}]) \\ \geq\ & 1 - \prod_j (1 - \frac{1}{2e^4} Prob[a_j \geq k/2]) \\ \geq\ & 1 - \prod_j (1 - \frac{1}{2e^{13}} Prob[Z \geq k/2]) \\ \geq\ & 1 - \left(1 - \frac{r}{2e^{13}d}\right)^{\frac{d}{r}} \geq 1 - e^{\frac{-1}{2e^{13}}}, \end{aligned}$$

where the last inequality holds for sufficiently large n. Thus, $E[\max\{a_i\}] \geq (1 - e^{\frac{-1}{2e^{13}}})(k/2) \geq (1 - e^{\frac{-1}{2e^{13}}})(\beta/8) = \Omega(\beta)$. Recalling that the expected optimal cost is at most $8e$, the theorem follows. $\blacksquare$

3 Routing

We show $n(\log n + \log r)$-node networks with n input nodes where a maximum edge congestion of $\sqrt{\frac{n}{r}}$ can be guaranteed by a particular r-strategy. Theorem 17 proves this to be at the least nearly optimal.

Theorem 15 *Divide all the input nodes into groups of r consecutive nodes. If each agent knows the destinations of the other input nodes in its group then we can route any $n \times n$ permutation on an $n(\log n + \log r)$-node network with maximum edge congestion $\sqrt{\frac{n}{r}}$.*

Proof. The network is derived from the Beněs network. We take the first $\log(n/r)/2$ levels of a $\log n$-dimensional Beněs network, then the middle $2 \log r$ levels of that network, then the next $\log(n/r)/2$ levels. As with global routing on a Beněs network, the bound on edge congestion is guaranteed even if there are two inputs for each source and two outputs for each destination. In the proof, we trace the selection of each path by describing the motion of a "packet" that moves along the path.

In every level, number each node from top to bottom $0, 1, ..., n-1$. The nodes in each level are divided into $\frac{n}{r}$ groups of r nodes. The groups in the first level (the input level) determine the knowledge graph — each group is a clique in the knowledge graph.

For the first $\log(n/r)/2$ levels, each input picks the greedy path, based on the first $\log(n/r)/2$ bits of the destination.

At this point, each packet reaches a place which differs with its input in only the first $\log(n/r)/2$ bits. There are $\frac{n}{r}$ $\log r$-dimensional Beněs networks in the middle. The source nodes of each such subnetwork (i.e., the nodes at level $\log(n/r)/2 + 1$ of the whole network) receive packets from at most $\sqrt{\frac{n}{r}}$ different cliques. Furthermore, the place of each packet within its clique has not changed (that is, the place of each packet agrees with its source in the last $\log r$ bits). Thus, each node has at most two packets from each clique.

Now we will use the next $2 \log r$ levels to route within each $r \times r$ network according to the last r bits of the destination. (to be explained later).

Afterwards, the place of each packet corresponds with its destination in the first $\log(n/r)/2$ bits and the last $\log r$ bits. If the bits are numbered from left to right, then the place of a packet can only disagree with its destination in bits $\log n/r/2 + 1$ through $\log n/r$. Thus, there are at most $\sqrt{n/r}$ packets per node. Each packet then takes the greedy path to its destination. The congestion never exceeds $\sqrt{n/r}$, since at each successive level, the upper bound on the congestion decreases by a factor of two.

So far, everything specified about the paths can be determined by each source node without any extra information besides the destination of its own inputs.

Now we have to explain how to do the routing in the middle $r \times r$ subnetworks. Now *source* and *destination* refer to the source and destination within the $r \times r$ network. We are guaranteed that for any such subnetwork, we only have $\sqrt{n/r}$ cliques routing simultaneously on that subnetwork. There are at most two packets per source from any single clique. There are at most $\sqrt{n/r}$ packets per destination from all cliques. Each clique determines the paths for all inputs in that clique so as to satisfy the conditions of Lemma 16 below. We then examine what happens when we have $\sqrt{n/r}$ cliques superimposed on the same $r \times r$ network.

In each level, number the nodes $0, 1, ..., r-1$ from top to bottom. Number levels from right to left so that the destination nodes are at level 1. For $j \in \{0, ..2^{\log r - k - 1} - 1\}$, define $S_i(j,k)$ to be the nodes in the i^{th} level numbered $2^{\log r - k + 1}x + j$, for all $x \in \{0, 1, .., 2^{k-1} - 1\}$. So, for any i, i' with $i > i'$ and any $k \geq i$, packets arriving at nodes in $S_i(j,k)$ can only reach level i' nodes in $S_{i'}(j,k)$. In particular, packets arriving in $S_i(j,i)$ can only reach level 1 nodes in $S_1(j,i)$. We have the following lemma:

Lemma 16 *Each clique can route the paths of its inputs so that*

1. *In the first half of any particular $r \times r$ Beněs network, each edge carries at most one path; and*

2. *In the second half of the network, the congestion along each edge coming into nodes in $S_i(j,i)$ from the previous level is the same (± 1).*

Proof. The proof is by induction on r. Consider the two $(r-1) \times (r-1)$ subnetworks in the middle. Suppose we manage to route the $r \times r$ problem so that each source sends one packet through the bottom network and one through the top network. And suppose that we manage to route the packets so that if a destination gets t packets, $t/2$ are routed through the bottom network and $t/2$ are routed through the top. (If t is not even, the number of packets routed through the top and the number routed through the bottom may differ by 1). Then we have managed to satisfy the constraints for the outer two level. By induction, we can satisfy the constraints for the inner levels.

So we just have to show that we can route the packets in that way. Make a bipartite graph - left vertices represent sources, right vertices represent destinations. An edge from a left vertex to a right vertex, represents a path routed from that source to that destination. The degree of each vertex on the left is two. The edges can be colored red and blue so that every vertex has half its edges red and half blue. The red edges represent paths through the upper subnetwork and the blue edges represent paths through the lower one. To see that the coloring of the edges can be done, split every vertex on the right with degree higher than two into vertices of degree two and at most one of degree one. Now combine pairs of degree one vertices so that the graph is 2-regular. By Hall's Theorem, the edges can be colored so that every vertex is incident to a red and a blue edge. When the vertices are recombined to get the original graph, if a vertex is incident to d edges, then at least $\lfloor d/2 \rfloor$ of the edges are colored with each color. ∎

We argue that if each clique can route its paths through each subnetwork so that conditions 1 and 2 are maintained, then the congestion is at most $\sqrt{n/r}$ even when we consider the congestion from all $\sqrt{n/r}$ cliques that are routing simultaneously on a particular subnetwork. In the first half of the network, each clique has only one path per edge. Thus, the total edge congestion is at most $\sqrt{n/r}$. In the second half of the network, paths that reach a node in $S_i(j,i)$ can only reach nodes in $S_1(j,i)$. If there are x paths coming into a node in $S_i(j,i)$ then property 2 guarantees that there are a total of $x2^{i-1}$ paths coming into all nodes in $S_i(j,i)$. These paths are all destined for nodes in $S_1(j,i)$. There can only be $2^{i-1}\sqrt{n/r}$ such paths because each destination gets only $\sqrt{n/r}$ paths from all cliques. Thus $x \leq \sqrt{n/r}$. ∎

Theorem 17 *For every n-input, n-output, N-node, degree-d network, for every deterministic routing strategy with degree-r knowledge graph,*

1. *there exists a permutation for which the maximum congestion at a node is at least $\frac{1}{2}\sqrt{\frac{n}{N}\frac{n}{(d+1)r}} - \frac{n}{2Nr}$;*

2. *there exists a permutation for which the maximum congestion at an edge is at least $\frac{1}{2d}\sqrt{\frac{n}{N}\frac{n}{r}}$.*

The proof is more or less a straightforward adaptation of the oblivious routing lower bounds of Borodin and Hopcroft [7] and of Kaklamanis, Krizanc and Tsantilas [14].

Proof. *Part 1.* Let S be an independent subset of size $\frac{n}{2r}$ in the knowledge graph. It has a neighbor set (in the knowledge graph) of size at most $\frac{n}{2}$. Fix the destinations of the sources in this neighbor set. We will ignore the congestion due to these paths. The algorithm for the inputs in S is now specified by a set of at least $\frac{n}{2}$ paths for each input — one to each remaining target.

Let $a = \frac{1}{2}\sqrt{\frac{n}{N}\frac{n}{(d+1)r}}$. We can assign to each of $t \geq \frac{n}{2} - da$ of the possible paths of an input node in S an internal node w, such that each assigned w is assigned for at least a paths and at most $(d+1)a = \frac{1}{2}\sqrt{\frac{n}{N}\frac{n(d+1)}{r}}$ paths. The reason is as follows. Let $u \in S$. Repeat the following process. Find a node w that is an internal node of at least a paths from u. Mark it and assign w to a of these paths. Remove the paths and repeat till no such node can be found. Now, for each of the paths that have not been assigned a node, trace the path from its destination to its source, and assign it the first marked node that it encounters (excluding its source or destination), if there is one. A path does not get assigned a node either if its destination is a neighbor of u, or if it passes through an unmarked neighbor of u. There are at most da such paths. The number of paths that get assigned any particular marked node w is at most the a paths that marked w, plus any path that passed through an unmarked neighbor of w. There are at most da of the latter.

The number of distinct nodes w assigned for a fixed input in S is at least $\frac{n/2-da}{(d+1)a} \geq \frac{n}{2(d+1)a} - 1$. There are $\frac{n}{2r}$ nodes in S, so a total of $\frac{n^2}{4(d+1)ra} - \frac{n}{2r}$ nodes w are hit. Since there are N nodes in the network, there is a node w that is hit by at least $\frac{n^2}{4N(d+1)ra} - \frac{n}{2Nr} = \frac{1}{2}\sqrt{\frac{n}{N}\frac{n}{(d+1)r}} - \frac{n}{2Nr}$ paths from different input nodes. Each input node that hits w can choose from among at least $\frac{1}{2}\sqrt{\frac{n}{N}\frac{n}{(d+1)r}}$ destinations that cause a hit on w, so there is a choice of distinct destinations for all input nodes that hit w.

Part 2. Let S be an independent set of size $t = n/2r$ in the knowledge graph. As in part 1, we fix the destinations of the sources in the neighborhood set of S and ignore the congestion due to the paths they choose. The destination of any source in S can be any of at least $n/2$ remaining output nodes in the network. An algorithm is specified by $nt/2$ paths. There are at least $t-1$ paths that end at any output node v (because v can also be a source node in S). Let $S(v)$ denote the set of edges in the network which have $k = \sqrt{\frac{n}{N}\frac{n}{r}}/2d$ or more paths ending in v passing through them. Let $S^*(v)$ be the set of nodes incident to edges in $S(v)$. Note that $|S^*(v)| \leq 2|S(v)|$. Also, $v \in S^*(v)$. Thus,

$$|S - S^*(v)| \leq (k-1)d|S^*(v)|.$$

(For each source u not in $S^*(v)$, follow its path to v until it hits a node in $S^*(v)$. The last edge in this path has less than k paths in it.) This gives us that

$$t \leq kd|S^*(v)| \leq 2kd|S(v)|.$$

Thus $t/2kd \leq |S(v)|$. Summing over all $n/2$ destinations v,

$$\sum_{v \in V} |S(v)| \geq \frac{nt}{4kd}.$$

Since there are $Nd/2$ edges in the network, there is some edge e for which $e \in S(v)$ for at least

$$\frac{nt/4kd}{Nd/2} = k$$

different values of v.

Select e and $v_1, \ldots, v_k$ such that $e \in S(v_i)$ for $1 \leq i \leq k$. This means that we can find $u_1, \ldots, u_k$ such that $u_i \neq u_j$ for $i \neq j$ and the paths from u_i to v_i all pass through e. ∎

The following is a lower bound for randomized oblivious routing strategies. The lower bound holds when the knowledge graph is a set of fixed r-cliques.

Theorem 18 *For every n-input, n-output, n-node, degree-d network, for every randomized routing strategy with a deterministic knowledge graph consisting of disjoint r-cliques, if $dr \leq n/2\log^4 n$, then there is a permutation for which the expected maximum congestion at a node is in $\Omega\left(\frac{\log(\frac{n}{r})}{\log\log(\frac{n}{r})}\right)$.*

Proof. The proof follows a lower bound on randomized oblivious routing due to Borodin et al. [9]. We show a probability distribution over permutations that beats every deterministic routing strategy. Let $t = n/r$. There are t r-cliques in the knowledge graph. Each node will be numbered by a pair (i,j), where i represents the name of the clique the node is in and j represents the name of the node within the clique. $1 \leq i \leq t$ and $1 \leq j \leq r$. We pick one of $t!$ permutations uniformly at random. Each permutation is specified by a permutation over the cliques. If clique i is mapped to clique k, then (i,j) has to connect to (k,j) for all $1 \leq j \leq r$. Thus each node has to connect to one of t destinations. For some valid source-destination pairs (u,v), we assign a node (called $V(u,v)$) which is an internal node in the path from u to v. $V(C,D)$ denotes the multiset of nodes $V(u,v)$, $u \in C$, $v \in D$, (u,v) a valid source-destination pair, and $V(u,v)$ exists. $|V(C,D)|$ denotes the size of the set, counting multiplicity.

Lemma 19 *Consider a clique C. The assignment can be picked so that*

1. *A node w appears in at most $(d+1)\log t$ multisets $V(C,D)$.*
2. *If a node w appears in a multiset $V(C,D)$, then it appears in $V(C,D)$ for at least $\log t$ different cliques D.*
3. $\sum_D |V(C,D)| \geq n - dr\log t$.

Proof. Consider all the paths from a clique C to all the destination cliques D. There are a total of n paths. (For each source-dest pair of cliques, there are r paths, and there are t destination cliques). Color each path so that the paths arriving at the same clique have the same color. This means that all the paths originating at a single source node have different colors. There are at most r paths colored with the same color. Now for each node that has at least $\log t$ paths of different colors going through it, mark the node and assign it $\log t$ paths of different colors. Then for each unassigned path, follow it from destination to source and assign it to the first marked node that it hits. A node will get at most $dr\log t$ paths of at most $d\log t$ colors this way. (This is because if a path gets assigned to a node, it came from an unmarked node. There are at most d unmarked nodes adjacent to a marked node. The paths that pass through an unmarked node have fewer than $\log t$ colors. There are at most r paths of a given color.)

How many paths don't get assigned? A path doesn't get assigned if it reaches its source node without hitting a marked node. There are r source nodes. How many make it back to a single source node u without getting assigned? If it doesn't get assigned, it passes through an unmarked neighbor of u just before it hits u. Call the neighbor v. All the paths going into u have different colors. If v is unmarked, than there are fewer than $\log t$ paths of different colors that pass through v. Thus there are fewer than $d \log t$ unassigned paths that reach u. ■

In what follows we denote by (C, D) the color of the r paths that lead from sources in C to destinations in D. Each of the t^2 combinations has a distinct color.

Now, execute the following procedure. Find a node w that is hit by a single color (C, D) at least $\log t$ times. Remove all paths from C and repeat.

If at least $t/2$ such nodes are found, then we have the following situation: There is a sequence $C_1, C_2, \ldots, C_{t/2}$ of source cliques, and a sequence $D_1, D_2, \ldots, D_{t/2}$ of destination cliques (not necessarily distinct), such that if D_i is assigned to C_i, there is a node that is $\log t$ congested. The probability that D_i is assigned to C_i is $1/t$. Conditioned upon D_1 not assigned to C_1, D_2 not assigned to C_2, ..., D_{i-1} not assigned to C_{i-1}, the probability that D_i is assigned to C_i is still at least $1/2t$. Therefore, with probability at least $1 - (1 - 1/2t)^{t/2} \geq 1 - e^{-1/4}$, there is a $\log t$ congested node.

Otherwise, we have removed at most $t/2$ source cliques. The remaining source cliques have the property that no node is hit more than $\log t$ times by the same color. From now on, we consider only the remaining source cliques.

Let E_{Cw} denote the expected number of paths from sources in C that pass through w. The assignment of $V(C, D)$ has the property that $\sum_w \sum_C E_{Cw} \geq (n - dr \log t)/2$, where the second sum is taken over the remaining source cliques C.

Suppose that for all w, $\sum_C E_{Cw} \leq \log t$ (Otherwise the lower bound on the expected congestion is established). Let $\delta = 1/8$. A node w is *good* if $\sum_C E_{Cw} \geq \delta$. The number of good nodes is at least $\frac{n/2 - \delta n - dr \log t/2}{\log t - \delta} \geq \frac{n}{8 \log n}$.

Fix a good node w. The assignment of $V(C, D)$ has the following property. For every clique C either $E_{Cw} = 0$, or $\frac{\log t}{t} \leq E_{Cw} \leq \frac{(d+1)\log^2 t}{t}$. To see the upper bound recall that none of the nodes are hit more than $\log t$ times by a single color $(C, \cdot)$. By claim 1 of Lemma 19, the upper bound follows. The lower bound follows from claim 2 of Lemma 19.

We want to lower bound the probability that h source cliques hit w for some $h < \log t$. Each hit corresponds to some clique C such that $E_{Cw} \geq \frac{\log t}{t}$. There are at least $\log t$ cliques D such that w is in $V(C, D)$. A hit by C

removes one destination clique D from the list of possible destinations of other source cliques. Consider the conditional expectation of the number of paths from sources in C that pass through w, conditioned upon at least h other source cliques having hit w. By the above arguments, this conditional expectation is at least $\frac{1}{t}$. Further notice that if $w \in V(C, D)$, it may appear in $V(C, D)$ at most $\log t$ times (because of our assumption on the remaining source cliques). If D is assigned to C, we will count this as only one hit, so the expected number of hits on w is bounded below by $\frac{1}{\log t} \sum_C E_{Cw}$. Also, the probability of a hit due to source C is at most $(d+1)\log t/t$.

We use the following probabilistic lemma, due to Borodin et al. [9].

Lemma 20 (Borodin et al.) *Let $x_1, \ldots, x_k$ be independent $0/1$ random variables. Let $p_i = Prob[x_i = 1]$, $\sum p_i = \sigma$ and $a \leq p_i \leq b$. Then, for any $Y \leq \frac{\sigma}{2eb\log(b/a)}$,*

$$Prob[\sum x_i > Y] \geq \left(\frac{\sigma}{2e\log(b/a)Y}\right)^Y .$$

In our case, $\sigma = \delta/\log t = 1/8\log t$, $a = 1/t$, $b = (d+1)\log t/t$.

According to the lemma, for n sufficiently large, the probability that node w is at least $Y = \frac{\log t}{8\log\log t}$ congested is at least $1/\sqrt{t} \geq \log t/t$, provided that the condition on Y holds. It can be easily verified that for sufficiently large n, the condition that $dr \leq n/2\log^4 n$ implies the required condition on Y.

To summarize, we have shown that one of the following properties holds:

1. There is a sequence of (not necessarily distinct) nodes $x_1, x_2, \ldots, x_{t/2}$ such that with constant probability at least one of them is at least $\log t$ congested; or

2. there is a node w whose expected congestion is at least $\log t$; or

3. there is a node w_1 that is at least $\log t/8\log\log t$ congested with probability at least $1/\sqrt{t} \geq \log t/t$.

If the first or second case holds, we are done. If the third case holds, we would like to construct a sequence of $t/\log t$ nodes $w_1, w_2, \ldots, w_{t/\log t}$ with the property that for every i, if $w_1, w_2, \ldots, w_{i-1}$ are less than $\log t/\alpha\log\log t$ congested, then w_i is at least $\log t/\alpha\log\log t$ congested with probability at least $\log t/t$, for some constant α. We construct such a sequence by repeating the entire argument above at most $t/\log t$ times. If $w_1, w_2, \ldots, w_i$, $i < t/\log t$, are less than $\log t/\log\log t$ congested, this constrains $t' = i \log t/\log\log t < t/\log\log t$ source cliques to their destination cliques. We remove these cliques, and can thus repeat the above discussion setting $t := t'' = t - t'$, and assigning new values $V(u, v)$ for the remaining cliques. Notice that $t'' \geq t(1 - 1/\log\log t)$. We get one of the following properties:

1. There is a sequence of (not necessarily distinct) nodes $y_1, y_2, \ldots, y_{t''/2}$ such that with constant probability at least one of them is at least $\log t'' \geq \log t - 1$ congested; or
2. there is a node w' whose expected congestion is at least $\log t'' \geq \log t - 1$; or
3. there is a node w_{i+1} that is at least $\log t''/8 \log\log t'' \geq (\log t - 1)/8 \log\log t$ congested with probability at least $1/\sqrt{t''} \geq \log t/t$.

All the probabilities and expectations are conditioned upon the assignment of destination cliques to the t' constrained source cliques.

In the first case, recall that it followed from each of the y's being hit by $\log t''$ paths of one pair of source-destination cliques. If we remove the conditioning upon the assignment of destination cliques to the t' constrained source cliques, this at most halves the probability of such a hit (because $t' < t/2$), so there would still be a constant probability for one of the y's to be at least $\log t - 1$ congested. In the second case, a similar argument shows that removing the condition at most halves the expected congestion at w' (using linearity of expectations). In the third case, we have constructed another node in the sequence $w_1, w_2, \ldots, w_{t/\log t}$, and may proceed to constructing the next one. $\blacksquare$

In the above proof, the permutation depends on the r-cliques. Assuming that $dr^3 \leq n/\log^3 n$, the lower bound in fact holds even when the agents can organize themselves into random r-cliques. The proof requires choosing uniformly at random an n-permutation of destinations rather than mapping cliques to cliques. We omit the proof.

4 Open Problems

We have given tight bounds, up to constant factors, for c_r, both in the deterministic and the randomized cases. It would be interesting to give better bounds for c_G in terms of other parameters of the graph G. Given G, what is the computational complexity of determining c_G? Our randomized upper bounds use shared coins. Can the same bounds be achieved with private coins? Our bounds for routing are far from being tight. Can one route deterministically on a high-degree network (where d is not a constant) achieving node or edge congestion close to the lower bounds given in Theorem 17? Does the randomized lower bound for node congestion hold for arbitrary knowledge graphs? Can a matching upper bound be achieved for large r?

5 Acknowledgements

We would like to thank the referees for their thorough reading and useful comments and suggestions. In particular, we would like to thank referee B for pointing out a flaw in our original proof of Lemma 10.

References

[1] A. Aggarwal, A.K. Chandra, M. Snir. On communication latency in PRAM computation. In *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, Santa Fe, 1989.

[2] N. Alon. Private Communication.

[3] B. Awerbuch, O. Goldreich, D. Peleg, R. Vainish. A Tradeoff Between Information and Communication in Broadcast Protocols. In *Proceeding of the 3rd Aegean Workshop on Computing*, July 1988.

[4] V. Beneš. *Mathematical Theory of Connecting Networks and Telephone Traffic*, Academic Press, New York, 1965.

[5] B. Bollobás. *Random Graphs*, Academic Press, 1985.

[6] A.H. Bond and L. Gasser, editors. *Readings in distributed artificial intelligence*, Morgan Kaufmann, 1988.

[7] A. Borodin, J. Hopcroft. Routing, merging, and sorting on parallel models of computation. *Journal of Computer and System Science*, 30(1):130-145, February 1985.

[8] A. Borodin, N. Linial, and M. Saks. An optimal on-line algorithm for metrical task systems. *Journal of the ACM*, 39:745–763, 1992.

[9] A. Borodin, P. Raghavan, B. Schieber, E. Upfal. How much can hardware help routing? In *Proceedings of the ACM 25th Symposium on the Theory of Computing*, San Diego, 1993.

[10] D. Culler, R. Karp, D. Patterson, A. Sahay, K.E. Schauser, E. Santos, R. Subramonian, T. von Eicken. LogP: Towards a realistic model of parallel computation. In *Proceedings of the 4th ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming*, San Diego, 1993.

[11] X. Deng, C.H. Papadimitriou. On the value of information. In *Proceedings of the 12th IFIPS Congress*, Madrid, 1992. Also in *Proceedings of the World Economic Congress*, Moscow, 1992.

[12] C. Dwork, M. Herlihy, O. Waarts. Contention in Shared Memory Algorithms. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, San Diego, 1993.

[13] R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:416-429,1969.

[14] C. Kaklamanis, D. Krizanc, T. Tsantilas. Tight bounds for oblivious routing. In *Proceedings of the 2nd ACM Symposium on Parallel Algorithms and Architectures*, July 1991.

[15] A.R. Karlin, M.S. Manasse, L. Rudolph, D.D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):70–119, 1988.

[16] C.H. Papadimitriou, M. Yannakakis. Towards an architecture-independent analysis of parallel algorithms. In *Proceedings of the 20th ACM Symposium on Theory of Computing*, 1988.

[17] C.H. Papadimitriou, M. Yannakakis. Linear programming without the matrix. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, San Diego, 1993.

[18] D.D. Sleator, R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communication of the ACM*, 28:202–208, February 1985.

[19] L.G. Valiant. A bridging model for parallel computation. *Communication of the ACM*, 33(8):103-111, August 1990.

[20] A. Waksman. A permutation network. *Journal of the ACM*, 15(1):159–163, January 1968.

[21] A.C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science*, 1977.