On the Value of Information in Coordination Games

(preliminary version)

Sandy Irani*

UC Irvine

Yuval Rabani†

ICSI, Berkeley

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 [10]. 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?

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 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* [17, 14]. 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 [10] and further discussed

by Papadimitriou and Yannakakis [16] 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 [10] is to model specific constraints in the availability of information and study the solution quality that can be obtained under these information regimes. This approach involves tedious and often ad-hoc analysis. In [16], 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. What is the most effective pattern of communication? To what extent is it useful for the agents to communicate about the piece of the input that each holds?

1.1 The Model

We consider optimization problems in the following context: A set of agents $A = \{a_0, a_1, \dots, a_{n-1}\}$ 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

*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: International Computer Science Institute, 1947 Center Street, Berkeley, CA 94704. Supported by a Rothschild post-doctoral fellowship. Email: rabani@icsi.berkeley.edu.

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.

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, 9, 11, 15, 18]. 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 [16], 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 connect-

ing 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 [5] for a comprehensive collection of papers in the field. The AI approach tends to be either qualitative or experimental. Recent theoretical results [10, 16] 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 [10]. 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, when G is 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 asymptotically 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 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 Benes network [4, 19] with maximum edge congestion of $\sqrt{\frac{n}{r}}$. The lower bounds on oblivious routing of [6, 13] can be modified to show lower bounds of $\frac{1}{d}\sqrt{\frac{n}{N}\frac{n}{r+1}}$ on edge congestion and $\frac{1}{2}\sqrt{\frac{n}{N}\frac{n}{dr}}$ 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. [8].

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. We prove the following theorem.

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

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 .

Theorem 2 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. Now instead, give $n/\sqrt{s}$ jobs to each of $\sqrt{s}$ of the agents

that sent a a job. Since there are n jobs total, the optimal cost is one. Agent a still gets $\sqrt{s}$ jobs.

Let $\phi(G) = m$. Partition the vertices into m cliques, $c_0, \dots, c_{m-1}$. 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: $a_0, \dots, a_{n-1}$. Let $d = \lfloor n/m \rfloor$. 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 of length. Let $s_0, \dots, s_{n-1}$ be the total length of the jobs in each group in decreasing order. (If the k^{th} group is empty, $s_k = 0$.) Clique c_i sends the jobs in group k to agent $a_{di+k \pmod{n}}$. Let $t(j, i)$ denote the total length of jobs that agent j receives from clique i . Notice that if $di \leq j < d(i+1)$ then clique i sends at least $t(j, i)$ jobs to each of $j - di + 1$ agents ($di, di+1, \dots, j$), clique $i-1 \pmod{m}$ sends at least $t(j, i-1 \pmod{m})$ jobs to each of $d+j-di+1$ agents, clique $i-2 \pmod{m}$ sends at least $t(j, i-2 \pmod{m})$ jobs to each of $2d+j-di+1$ agents, etc.

Suppose that agent j gets assigned jobs whose length sums to T . Let i_j be the number that satisfies $di_j \leq j < d(i_j+1)$ and t_i be $t(j, i_j - i \pmod{m})$. Let $T_1 = \sum_{i=0}^{\sqrt{m}-1} t_i$, and $T_2 = \sum_{i=\sqrt{m}}^{m-1} t_i$. Then $T = T_1 + T_2 \leq 2 \max\{T_1, T_2\}$. The optimal strategy sends at least $\max_{0 \leq i < \sqrt{m}} t_i$ jobs to some agent, so if $T_1 \geq T_2$,

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

The total length of all the jobs in the system is at least $\sum_{i=\sqrt{m}}^{m-1} d i t_i \geq d\sqrt{m} \sum_{i=\sqrt{m}}^{m-1} t_i$. Thus, the optimal solution sends jobs of length at least

$$\frac{d\sqrt{m} \sum_{i=\sqrt{m}}^{m-1} t_i}{n} \geq \frac{\sum_{i=\sqrt{m}}^{m-1} t_i}{\sqrt{m}} = \frac{T_2}{\sqrt{m}}$$

to some agent. Thus, if $T_2 \geq T_1$, $\frac{T}{T_2/\sqrt{m}} \leq \frac{2T_2}{T_2/\sqrt{m}} = 2\sqrt{m}$. ■

Remark. The upper bound can be improved to $\sqrt{2m}$ 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, the greedy algorithm which assigns each job in turn to the group with minimum total length comes within a factor of 2 of the optimal solution [12]. Thus the above upper bound can be achieved by polynomial time bounded agents with an extra factor of 2 in the ratio.

Proof of Theorem 1. 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 $\lceil n/(r+1) \rceil$ complete graphs, one on each of the subsets. ■

Claim 3 shows that the upper bound given by Theorem 2 is not tight. 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 $G(n, \frac{1}{2})$, then with high probability, $c_G = O(n^{\frac{1}{3}} \log n)$.*

The proof will appear in the full version of the paper.

We show the following upper bound on randomized strategies.

Theorem 4 *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. Let $m = \lceil \frac{n}{r} \rceil$. The knowledge graph we use is a set of m disjoint r -cliques. We assume that the agents within a clique know the set of jobs assigned to the other agents in the cliques as well as their random bits. Thus, each clique operates as a single agent. Each agent will be indexed by a pair (i, j) where $0 \leq i \leq m-1$ and $0 \leq j \leq r-1$. i indicates the clique to which the agent belongs and j indicates the place within the clique.

Each clique c_i divides its tasks into at most n groups such that the maximum length of a group's tasks is minimized. Then it sorts the groups in decreasing order of length. Each group will be indexed by a triplet (i, j, l) where $0 \leq i, l \leq m-1$ and $0 \leq j \leq r-1$. The third index denotes the clique where the jobs originate. Let $s(i, j, l)$ be the total length of the jobs in group (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 $j > j'$. For each $i \in \{0, \dots, m-1\}$, clique c_i draws $k \in \{0, \dots, m-1\}$ uniformly at random. Then for all $j \in \{0, \dots, r-1\}$, it sends all the jobs in group (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 $E[D] \in O((\log m / \log \log m) OPT)$.

Consider a related problem: we have m agents. Agent l gets a set of m jobs of lengths $s(0, 0, l), s(1, 0, l), \dots, s(m-1, 0, l)$. (Some of these could be of length 0). Let OPT' be the cost when the jobs are optimally distributed among the m 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 [10] gives that $E[D'] \in O((\log m / \log \log m) OPT')$. 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 groups:

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

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

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

1. $OPT \geq \max\{MAX, \frac{T_2}{m}\};$
2. $OPT' \leq MAX + \frac{T_1+T_2}{m}.$

The theorem follows from the two claims because

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

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

$$\begin{aligned} OPT &\geq \frac{1}{n} \sum_{j=0}^{r-1} \sum_{i=0}^{m-1} \sum_{l=0}^{m-1} s(i, j, l) \\ &\geq \frac{r}{n} \sum_{i=1}^{m-1} \sum_{l=0}^{m-1} s(i, 0, l) \\ &\geq \frac{rT_2}{n} = \frac{T_2}{m}. \end{aligned}$$

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 $(T_1 + T_2 + m \cdot disc)/m$ is an upper bound on the optimal cost, the claim follows. ■

Theorem 4 is tight up to a constant factor, as the following theorem shows.

Theorem 5 *If $\sqrt[r(n)]{\frac{r(n)}{n}} \rightarrow 0$ as $n \rightarrow \infty$, 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 at least $\frac{1}{7000} \frac{\log(\frac{n}{r})}{\log \log(\frac{n}{r})}$.*

Remark. If $\sqrt[r(n)]{\frac{r(n)}{n}} \geq \epsilon > 0$ 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 5. We need to consider only r -regular knowledge graphs (i.e., with *out-degree* r). Applying von Neumann's minimax principle (see [20, 7]), we show a probability distribution over inputs which beats every deterministic algorithm. This distribution

is such that every agent receives d jobs with probability p and 0 jobs otherwise. We use $d = \sqrt[4]{r^3 n}$ and $\frac{4}{d} < p < \frac{4e(r+1)}{dr}$, so that $p(1-p)^r dr/(r+1) = 4$. Notice that $d \in \omega(r) \cap o(n)$.

The expected cost of the optimal algorithm is at most $8e$. Let $\beta = \log(\frac{n}{r})/\log\log(\frac{n}{r})$. Our goal is to show that the expected cost of any deterministic r -strategy is in $\Omega(\beta)$.

We need the following lemma.

Lemma 6 *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,*

$$\text{Prob}[Z \geq \frac{\log(d/r)}{\log\log(d/r)}] \geq \frac{r}{d}.$$

Proof. Let $k = \log(\frac{d}{r})/\log\log(\frac{d}{r})$. Let $\lambda = E[Z] = p'd/4(r+1) = 1$.

$\text{Prob}[Z = k]$ (which is a lower bound for $\text{Prob}[Z \geq k]$) can be estimated using the Poisson approximation to the binomial distribution. It gives

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

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

We have, for sufficiently large n ,

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

The condition that $\sqrt{\frac{r(n)}{n}} \rightarrow 0$ as $n \rightarrow \infty$, implies that $r(n) \in o(d(n))$. Also, $\log(\frac{d}{r})/\log\log(\frac{d}{r}) \geq \log(\frac{n}{r})/4\log\log(\frac{n}{r})$. So we can conclude that $\text{Prob}[Z \geq \frac{\beta}{4}] \geq \frac{r}{d}$.

A vertex of G is *chosen* if it gets jobs. A vertex is *isolated* if it is chosen and none of its neighbors are chosen.

Lemma 7 *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 β . Assume $\sum_{v \in A} W_v \geq \frac{d}{4}$. Let Y be the sum of the weights of the isolated vertices in A . Let Z be as in Lemma 6. Then, for every $k \leq \sqrt{\frac{d}{8r\beta}}$,*

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

Proof. Let $s = d/4$. We may assume that $\sum_{v \in A} W_v = s$, otherwise we reduce weights and remove 0-weighted vertices till 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 $Y_1, \dots, Y_s$. $Y = \sum Y_j$. Now consider a particular setting where A has exactly $d/4$ vertices, each of weight 1, and the subgraph of G induced by A consists of disjoint $(r+1)$ -cliques. Let $Z_1, \dots, Z_s$ be the indicator variables in this case. Clearly, the distributions of Z and of $\sum Z_j$ are identical. We wish to show that for any $k \leq \sqrt{d/8r\beta}$,

$$\text{Prob}\left[\sum_{j=1}^s Y_j \geq k\right] \geq \frac{1}{2e} \text{Prob}\left[\sum_{j=1}^s Z_j \geq k\right].$$

Define

$$S_k = E \left[\sum_{X \subset \{1, \dots, s\}, |X|=k} \prod_{j \in X} Y_j \right].$$

Similarly

$$\hat{S}_k = E \left[\sum_{X \subset \{1, \dots, s\}, |X|=k} \prod_{j \in X} Z_j \right].$$

First we prove that $S_k \geq \hat{S}_k/e$.

Examine a fixed k -set X of $\{1, \dots, s\}$. Let $V(X)$ be the set of vertices associated with the random variables Y_j , $j \in X$. When does $\prod_{j \in X} Y_j = 1$? This happens when $V(X)$ forms an independent set, all the vertices in $V(X)$ are chosen and all the vertices in the neighborhood set of $V(X)$ are not chosen. Notice that $|V(X)| \leq k$ and the neighborhood set of $V(X)$ has at most kr vertices. Thus if $V(X)$ forms an independent set, then $\prod_{j \in X} Y_j = 1$ with probability at least $p^k(1-p)^{rk}$.

Now let's look at the Z_j s. Let $\hat{V}(X)$ be the set of vertices associated with the random variables $\{X_j \mid j \in X\}$. If X is a k -set, then $|\hat{V}(X)| = k$. If $\hat{V}(X)$ forms an independent set, then the neighborhood set of $V(X)$ has size rk (because the graph is composed of disjoint $(r+1)$ -cliques). Thus if $\hat{V}(X)$ 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 such that $V(X)$ is an independent set is at least $1/e$ times the number of k -sets such that $\hat{V}(X)$ is an independent set. We do this by picking X at random and showing that $\text{Prob}[V(X) \text{ is independent}] \geq e^{-1}$. First we form a graph G' . Each vertex in G' corresponds to a Y_i . There is an edge between Y_i and Y_j if their associated vertices are adjacent in G . The degree in G' is at most $r\beta$. What is the probability that

a random k -set in G' is independent? We pick a random k -set, vertex by vertex. Let v_i denote the i^{th} vertex that is picked and V_i denote $\{v_1, \dots, v_i\}$. The neighborhood set of V_i is denoted $N(V_i)$.

Thus, we have:

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

Therefore, we conclude that $S_k \geq \hat{S}_k/e$. We now complete the proof that $\text{Prob}[Y \geq k] \geq \text{Prob}[Z \geq k]/2e$. We use the principle of inclusion-exclusion:

$$\text{Prob}[Y \geq k] = S_k - S_{k+1} + S_{k+2} - \dots + (-1)^{n-k} S_n.$$

Therefore, $\text{Prob}[Y \geq k] + \text{Prob}[Y \geq k+1] = S_k$. Thus,

$$\begin{aligned} \text{Prob}[Y \geq k] &\geq \frac{1}{2}(\text{Prob}[Y \geq k+1] + \text{Prob}[Y \geq k]) \\ &\geq \frac{1}{2}S_k \geq \frac{1}{2e}\hat{S}_k \\ &\geq \frac{1}{2e}\text{Prob}[Z \geq k]. \quad \blacksquare \end{aligned}$$

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 β or more. Then, we prove the theorem for graphs H that do not fall into either of these categories.

Lemma 8 *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 $u_1, u_2, \dots, u_k$ be the neighbors of v on the left. For $i = 1, 2, \dots, k$, 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 gets is $p(1-p)^r \sum W_i \geq p(1-p)^r \beta d \geq 4\beta$. $\blacksquare$

Lemma 9 *If there are at least d left vertices in H with an edge of multiplicity β or more, then the expected maximum number of jobs an agent gets is at least $\beta/2e^2$.*

Proof. There are d vertices such that if any of them is isolated, some agent has a load of at least β . Consider a subset A of $d/4$ of those vertices. Assign unit weight to every vertex in A . Applying Lemma 7 to this setup, the probability that there is at least one isolated vertex in A is at least $\text{Prob}[Z \geq 1]/2e$. Since $\text{Prob}[Z \geq 1] \geq e^{-1}$, the expected maximum load is at least $\beta/2e^2$. $\blacksquare$

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

We need the following lemma.

Lemma 10 *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 have minimum root degree of $d/2 - cd/r \geq d/4$ and they are vertex disjoint. $\blacksquare$

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 $\mathcal{A} = A_1, A_2, \dots, A_{d/r}$. 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 β .

The weight of a set is the sum over all the vertices in the set of the weight of each vertex. Each set has weight 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 $(a_1 < k) \wedge (a_2 < k) \wedge \dots \wedge (a_j < k)$.

Lemma 11 *Let $k \leq \log(\frac{n}{r}) / \log \log(\frac{n}{r})$. Let $1 \leq j \leq d/r$. If $\text{Prob}[E_{j-1}] \geq 1 - \frac{1}{2e}$ then*

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

Proof. Let $A = A_1 \cup A_2 \cup \dots \cup A_{j-1}$. Let $E = E_{j-1}$. For each vertex $v \in A_j$ we introduce W_v indicator variables as in the proof of Lemma 7. Let $Y_1, \dots, Y_s$ be those variables. Define

$$S_k = \sum_{I \subseteq \{1, \dots, s\}, |I|=k} \prod_{j \in I} Y_j.$$

Using the arguments from the proof of Lemma 7, it is sufficient to prove that $E[S_k \mid E] \geq E[S_k]/2e$. Fix an arbitrary k -set $I \subseteq \{1, \dots, s\}$. Let X be the set of vertices associated with the random variables Y_j , $j \in I$. Notice that $|X| \leq k$. We need to prove that $\text{Prob}[X \text{ is isolated} \mid E] \geq \text{Prob}[X \text{ is isolated}]/2e$.

The event that X is isolated is equivalent to the fact that X is chosen and $N(X)$ is not chosen. Therefore,

$$\begin{aligned} & \text{Prob}[X \text{ is isolated}] \\ &= \left(\frac{1}{d}\right)^{|X|} \text{Prob}[N(X) \text{ is not chosen}]. \end{aligned}$$

Also,

$$\begin{aligned} & \text{Prob}[X \text{ is isolated} \mid E] \\ &= \left(\frac{1}{d}\right)^{|X|} \text{Prob}[N(X) \text{ is not chosen} \mid E]. \end{aligned}$$

Since

$$\begin{aligned} \text{Prob}[N(X) \text{ is not chosen}] &= \left(1 - \frac{1}{d}\right)^{|N(X)|} \\ &\geq \left(1 - \frac{1}{d}\right)^{kr} \geq e^{-1}, \end{aligned}$$

we have that

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

We can now complete the proof of the theorem. Set $k = \beta/4$. The condition of Lemma 7 holds for sufficiently large n . If there exists j , $1 \leq j \leq d/r$, for which $\text{Prob}[E_j] < 1 - 1/2e$, then the expected maximum load is at least $\beta/8e$. Otherwise, we may use Lemma 11 to get

$$\begin{aligned} & \text{Prob}[\max a_j \geq k] \\ &= 1 - \text{Prob}[\forall j, a_j < k] \\ &= 1 - \prod_j \text{Prob}[a_j < k \mid E_{j-1}] \\ &= 1 - \prod_j (1 - \text{Prob}[a_j \geq k \mid E_{j-1}]) \\ &\geq 1 - \prod_j (1 - \frac{1}{4e} \text{Prob}[a_j \geq k]) \\ &\geq 1 - \prod_j (1 - \frac{1}{8e^2} \text{Prob}[Z \geq k]) \\ &\geq 1 - \left(1 - \frac{r}{8de^2}\right)^{\frac{d}{r}} \geq 1 - e^{-\frac{\beta}{8e^2}}, \end{aligned}$$

where the last inequality holds for sufficiently large n . Thus, $E[\max\{a_i\}] \geq (1 - e^{-\frac{\beta}{8e^2}})(\beta/4) \geq \beta/300$. 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 14 proves this to be at the least nearly optimal.

Theorem 12 *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 input node 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 $0, 1, \dots, n-1$. The nodes in each level are divided into $\frac{n}{r}$ groups of r nodes.

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 groups. Furthermore, the place of each packet within its group 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 group.

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 $\log n/r + 1$ 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 $\sqrt{n/r}$ groups routing simultaneously on that subnetwork. There are at most two packets per source from any single group. There are at most $\sqrt{n/r}$ packets per destination from all groups. We examine the routing pattern for each group individually and then analyze what happens when we have $\sqrt{n/r}$ groups superimposed on the same $r \times r$ network.

In each level, number the nodes $0, 1, \dots, r-1$ from top to bottom. Number levels from right to left so that the destination nodes are at level 1. For $j \in \{0, \dots, 2^{n-k-1} - 1\}$, define $S_i(j, k)$ to be the nodes in the i^{th} level numbered $2^{r-k+1}x + j$, for all $x \in \{0, 1, \dots, 2^{k-1} - 1\}$. We have the following lemma:

Lemma 13 *Each group can route the packets so that*

1. *In the first half of the Beněs network, there is at most one packet per edge; and*
2. *In the second half of the network, the congestion along each edge coming into $S_i(j, i)$ 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, that's where the ± 1 comes from). 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. Well, make a bipartite graph - left vertices represent sources, right vertices represent destinations. An edge from a left vertex to a right vertex, represents a packet 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 group can route its packets 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 $\sqrt{n/r}$ groups that are routing simultaneously on a particular subnetwork. In the first half of the network, each group has only one packet per edge. Thus, the total edge congestion is at most $\sqrt{n/r}$. In the second half of the network, packets that reach a node in $S_i(j, i)$ can only reach nodes in $S_1(j, i)$. If there are x packets coming into a node in $S_i(j, i)$ then property 2 guarantees that there are a total of $x2^{i-1}$ packets coming into nodes in $S_1(j, i)$. These packets are all destined for nodes in $S_1(j, i)$. There can only be $2^{i-1}\sqrt{n/r}$ such packets because each destination gets only $\sqrt{n/r}$ packets from all groups. Thus $x \leq \sqrt{n/r}$. ■

Theorem 14 *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 $\frac{1}{2}\sqrt{\frac{n}{N}\frac{n}{dr}}$;*
2. *there exists a permutation for which the maximum congestion at an edge is at least $\frac{1}{d}\sqrt{\frac{n}{N}\frac{n}{r+1}}$.*

The proof is more or less a straightforward adaptation of the oblivious routing lower bounds of Borodin and Hopcroft [6] and of Kalamanis, Krizanc and Tsantilas [13]. It will appear in the full version of the paper.

Theorem 15 *Let $N = n \log^c n$. 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 $dr \leq n/2 \log^{(c+4)} n$, 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. [8]. 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 at random. Each permutation sends packets from one clique to a random clique. If clique i is mapped to clique k , then (i, j) sends a packet to (k, j) for all $1 \leq j \leq r$. Thus each node sends a packet to one of t destinations. For every valid source-destination pair (u, v) , we assign a node (called $V(u, v)$) in which is an internal node in the path from u to v . Thus, if a clique C is assigned to clique D , then $V(C, D)$ is a multiset. $|V(C, D)|$ denotes the size of the set, counting multiplicity.

Lemma 16 Consider a clique C . The assignment can be picked so that

1. A node w appears in at most $(dr + 1) \log t$ multisets $V(C, D)$ (counting multiplicity).
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 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, then 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 . ■

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 $\sum_w \sum_C E_{Cw} \geq n - dr \log n$.

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

Assume at first that $r \leq \sqrt{\frac{n}{2 \log^4 n}}$. Suppose that for all w , $\sum_C E_{Cw} \leq \log t$ (Otherwise the lower bound on the expected congestion is established). Let $\delta = n/4N$. A node w is *good* if $\sum_C E_{Cw} \geq \delta$. The number of good nodes is at least $\frac{n - \delta N - dr \log t}{\log t - \delta} \geq \frac{n}{2 \log n}$.

If there exists a good node w that is assigned to at least $\log t$ paths of each of at least $\log t$ different colors, then the probability that w is at least $\log t$ congested is at least $\frac{\log t}{t}$.

Otherwise, the assignment of $V(C, D)$ has the following property. For every clique C either $E_{Cw} = 0$, or $\frac{\log t}{t} \leq E_{Cw} \leq \frac{((d+1) \log t + r) \log t}{t}$.

We want to lower bound the probability that h 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 up to h other source cliques having hit w . By the above arguments, this conditional expectation is at least $\frac{1}{t}$.

We use the following probabilistic lemma, due to Borodin et al. [8].

Lemma 17 (Borodin et al.) Let $x_1, \dots, x_k$ be independent 0/1 random variables. Let $p_i = \text{Prob}[x_i = 1]$, $\sum p_i = \delta$ and $a \leq p_i \leq b$. Then, for any $Y \leq \frac{\delta}{2eb \log(b/a)}$,

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

In our case, $\delta = n/4N$, $a = 1/t$, $b = ((d+1) \log^2 t + r \log t)/t$. Therefore, the condition required is that $dr + r^2 \leq n^2/N \log^4 n$. Since $r \leq \sqrt{\frac{n^2}{2N \log^4 n}}$, then the condition is that $dr \leq n^2/2N \log^4 n$. According to the lemma, the probability that node w is $\frac{\log t}{2(c+2) \log \log t}$ congested is at least $1/\sqrt{t} \geq \log t/t$. Now we have to show that this bound holds for the next $t' = \frac{t}{\log t}$ nodes like w that we consider. The basic point here is that if fewer than t' nodes are not congested, then this constrains $(t' - 1)(\log t / \log \log t)$ source cliques to their destination cliques. Even if we remove the traffic from these source-destination pairs, the total expected congestion remains $\Omega(n)$. There are still $\Omega(n/\log n)$ good nodes, where "good" is defined in terms of the remaining congestion.

Now, consider the case where $r > \sqrt{\frac{n^2}{2N \log^4 n}}$. Execute the following procedure. Find a good node that is hit by a single color at least $\log t$ times. Remove the node and color and repeat. If at least $t/2$ such nodes are found, then since the probability of each such node to be at least $\log t$ congested is at least $1/t$, there is a constant probability that the congestion in at least one of the nodes will be at least $\log t$. Otherwise, we have removed at most $t/2 \leq \sqrt{2N \log^4 n}/2$ good nodes and $t/2$ colors, and the remaining good nodes have the property that no remaining color hits them more than $\log t$ times, so, for each such node w , $\frac{\log t}{t} \leq E_{Cw} \leq \frac{(d+1)\log^2 t}{t}$, and the previous arguments hold under the assumption that $dr \leq n^2/N \log^4 n$. ■

In the above proof, the permutation depends on the r -cliques. Assuming that $dr^3 \leq n/\log^3 n$, We can show that the lower bound in fact holds even when the agents can organize themselves into random r -cliques. The proof will appear in the full version of the paper.

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. Beněš. *Mathematical Theory of Connecting Networks and Telephone Traffic*, Academic Press, New York, 1965.
- [5] A.H. Bond and L. Gasser, editors. *Readings in distributed artificial intelligence*, Morgan Kaufmann, 1988.
- [6] 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.
- [7] A. Borodin, N. Linial, and M. Saks. An optimal on-line algorithm for metrical task systems. In *Proceedings of the 19th ACM Symposium on Theory of Computing*, 1987.
- [8] A. Borodin, P. Raghavan, B. Schieber, E. Ufal. How much can hardware help routing? In *Proceedings of the ACM 25th Symposium on the Theory of Computing*, San Diego, 1993.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] R.L. Graham. Bounds for certain multiprocessing anomalies. *Bell Systems Technical Journal*, 45:1563-1581, 1966.
- [13] 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.
- [14] A.R. Karlin, M.S. Manasse, L. Rudolph, D.D. Sleator. Competitive snoopy caching. *Algorithmica*, 3(1):70-119, 1988.
- [15] 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.
- [16] C.H. Papadimitriou, M. Yannakakis. Linear programming without the matrix. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, San Diego, 1993.
- [17] D.D. Sleator, R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communication of the ACM*, 28:202-208, February 1985.
- [18] L.G. Valiant. A bridging model for parallel computation. *Communication of the ACM*, 33(8):103-111, August 1990.
- [19] A. Waksman. A permutation network. *Journal of the ACM*, 15(1):159-163, January 1968.
- [20] A.C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science*, 1977.