Fairness in Scheduling

(Extended Abstract)

Miklos Ajtai* James Aspnes† Moni Naor‡ Yuval Rabani§ Leonard J. Schulman¶
Orli Waarts‖

Abstract

On-line machine scheduling has been studied extensively, but the fundamental issue of fairness in scheduling is still mostly open. In this paper we explore the issue in settings where there are long living processes which should be repeatedly scheduled for various tasks throughout the lifetime of a system. For any such instance we develop a notion of *desired* load of a process, which is a function of the tasks it participates in. The *unfairness* of a system is the maximum, taken over all processes, of the difference between the desired load and the actual load.

An example of such a setting is the *carpool problem* suggested by Fagin and Williams [17]. In this problem, a set of n people form a carpool. On each day a subset of the people arrive and one of them is designated as the driver. A scheduling rule is required so that the driver will be determined in a 'fair' way.

We investigate this problem under various assumptions on the input distribution. We also show that the carpool problems can capture several other problems of fairness in scheduling.

1 Introduction

1.1 Our results

Consider the following *edge orientation problem*: on a set of n nodes labeled $\{1, \ldots, n\}$ there is a (possibly infinite) sequence of edges, i.e. pairs of nodes. Undirected edges arrive one by one, and each edge should be oriented upon its arrival. The goal is devise a method of orienting the edges so that in every node at every point in time the difference between the indegree and outdegree is as small as possible. (Note that it can be shown that for every sequence there exists an orientation such that at any point along the sequence the maximum difference is 1.)

The problem comes in three flavors:

1. Find a *deterministic* rule for orienting the edges and analyze it on the the worst input sequence.

2. Suggest a rule and analyze it under the some assumption on the distribution of the sequence, in particular that each edge in the sequence is chosen uniformly from all possible edges and independently of the rest of the sequence.

3. Suggest a *randomized* rule for orienting the edges and analyze its expected performance on the worst sequence.

The greedy algorithm is the one where an edge is oriented from the node with the smaller difference between the outdegree and indegree to the one with the larger difference. In the deterministic version of the rule ties are broken according to the lexicographic order. In the randomized version of the rule ties are broken at random.

We address the three flavors of the problem and obtain the following results:

1. The deterministic performance is $n/2$: there is a method (the greedy algorithm) that achieves this bound and for any deterministic rule there is a sequence where this difference will occur. (The lower bound is from [17]).

2. The expected difference of the greedy algorithm on the uniform distribution on the edges is $\Theta(\log \log n)$ and we derive a complete description of the process

*IBM Research Division, Almaden Research Center. E-Mail: ajtai@almaden.ibm.com

†Dept. of Computer Science, Yale University. Supported by NSF grant CCR-9410228. During part of this research this author was visiting IBM Almaden Research Center. E-Mail: aspnes@cs.yale.edu

‡Incumbent of the Morris and Rose Goldman Career Development Chair, Dept. of Applied Math and Computer Science, Weizmann Institute, Israel. Research supported by an Alon Fellowship and a grant from the Israel Science Foundation administered by the Israeli Academy of Sciences. E-Mail: naor@wisdom.weizmann.ac.il

§This work was done while the author was at ICSI, Berkeley, and at the Lab. for Computer Science, MIT. Work at ICSI supported by a Rothschild postdoctoral fellowship. Work at MIT supported by ARPA/Army contract DABT63-93-C-0038. Present address: Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A4, E-mail: rabani@cs.toronto.edu.

¶Computer Science Division, U.C. Berkeley. Work supported by an NSF postdoctoral fellowship. E-Mail: schulman@cs.berkeley.edu

‖Computer Science Division, U.C. Berkeley. Supported by an NSF postdoctoral fellowship. Part of this work was done while this author was at IBM Almaden Research Center. E-Mail: waarts@cs.berkeley.edu

in this case. This is the main technical contribution of the paper. In this abstract we show in detail only a proof of an $O(\log n)$ upper bound. We give an overview of the $\Theta(\log \log n)$ proof.

3. There is a rule (local greedy) with expected performance $O(\sqrt{n \log n})$ on any sequence. The lower bound is $\Omega(\sqrt[3]{\log n})$.

In addition we investigate the relationship between the edge orientation problem and the *vector rounding problem*. In this problem we are given a real matrix column by column and should produce an integer matrix so that each column in the output matrix is a rounding of the corresponding column in the input matrix that preserves the sum. The goal is to minimize the maximum over all rows of the difference between the sum of the row in the integer and real matrix. (A formal definition can be found in Section 4.) We show:

4. A general transformation from the vector rounding problem to the edge orientation problem, at the price of doubling the expected difference. The transformation applies to both deterministic and randomized algorithms.

As it turns out, problems in fairness of scheduling can be reduced to the vector rounding problem.

1.2 Motivation and Applications

On-line machine scheduling has been studied extensively (see e.g. [4, 5, 6, 8, 15, 18, 15, 22]), but the issue of fairness in job allocation has usually not been considered quantitatively (however, see [3, 13, 16, 19, 20]). In a typical on-line scheduling problem, there are n machines and a number of separate jobs; the jobs arrive one by one, and each job must be assigned to exactly one of the machines, thereby increasing the *load* on this machine by an amount that depends on both the job and the machine. The goal of the scheduling problems studied in all the above literature is to minimize the maximum machine load. The situation in which this model seems most applicable is if all machines have one owner that wishes to optimize their utilization. If the machines have different owners, then fairness in allocation may be an additional, or primary, parameter to be optimized by the scheduler; for instance the dispatcher for a number of independently owned taxicabs.

Assuming that machines (or their owners) are reluctant (or eager) to do the required jobs, a "fair" rule, which takes into account the benefit to each machine (owner) of performing each task, must be applied. Thus, when faced with such a problem we should define the *desired* load of a machine (the fair share) and then suggest an algorithm for scheduling the jobs that tries to give each machine a number of jobs corresponding to its fair share.

An interesting property of the results we obtain in studying "fair" scheduling, is that there are scheduling protocols for which the discrepancy between the loads of the machines can be bounded in terms of functions only of the number of the machines, with no dependence on the elapsed time.

We elaborate here on two cases where fairness is desired and see the connections with the edge orientation problem mentioned above.

The Carpool Problem The issue of fairness in scheduling was first isolated by Fagin and Williams [17], who abstracted it to what they call the *carpool* problem. A rough quotation from [17]: "Suppose that n people, tired of spending their time and money in gasoline lines, decide to form a carpool. Each day a subset of these people will arrive and one of them should drive. A scheduling algorithm is required for determining which person should drive on any given day. The algorithm should be perceived as fair by all members so as to encourage their continued participation."

The first question is how to define fairness. If the driver were not a member of the group, but a hired driver, then the meaning of fairness would be clear: The professional driver charges a fixed price for every ride, and each day the people that show up split the price of the driver equally among them. When the driver is just one of the set of people that show up, this reasoning leads immediately to the following definition of fairness given in [17]: If on a certain day, d people show up, each of them owes the driver $1/d$ of a ride. The *unfairness* of the algorithm at a certain point of the execution is defined as the maximum number of owed rides that anybody has accumulated up to that point or that anybody owes the rest of the group at that point. We will evaluate the algorithms according to the expected value of their unfairness when computed throughout the execution, and refer to it as *expected unfairness*.

Fagin and Williams proposed a natural algorithm for this problem, which we call the *global greedy* algorithm. When a set of people shows up, the one to drive will be the one that is currently the poorest. Ties are broken arbitrarily. They analyzed the performance of this algorithm assuming the worst case sequence of requests. In this setting they showed that the unfairness of this algorithm is bounded above by a number that is exponential in the number of people, but independent in the number of days, and mentioned that Coppersmith managed to reduce this upper bound to linear (however, this proof is lost [14]). Finally, in collaboration with Coppersmith they showed a linear lower bound on the unfairness in this setting.

We view the carpool problem as a game played be-

tween the carpool members and an adversary scheduler, which determines which people show up on each day. There are three main types of adversaries treated in the literature: the *adaptive*, the *oblivious*, and the *uniformly random*, where the distinction is made according to the way the adversary determines which people show up. The adaptive adversary constructs its schedule on the fly, making decisions that may depend on the whole previous history of the game. The oblivious adversary must fix its schedule before the game starts, though it may choose this schedule based on knowledge of the algorithm. The uniformly random adversary schedules people each day independently and uniformly at random.

Observe that the edge orientation problem is simply a special case of the carpool problem, restricted to two people arriving each day. (Though notice that the unfairness as measured in the edge orientation problem is double that as measured in the 2-person carpool problem.) (For this reason we shall refer to the performance of an algorithm for the edge orientation problem as its *unfairness*.) On the other hand, the (general) carpool problem is a special case of the vector rounding problem: each participant corresponds to a row, each day corresponds to a column; the ith entry of the jth column is 0 if the ith participant did not show up on the jth day, and is $\frac{1}{d_j}$ if d_j participants, including himself, did show up.

Therefore, an immediate byproduct of the results mentioned above on the edge orientation problem and of the general transformation to the vector rounding problem is that the general carpool problem has unfairness of $\Theta(n)$ against an adaptive adversary, and expected unfairness of $O(\sqrt{n \log n})$ against an oblivious adversary. (In fact, we also show directly that the natural greedy algorithm for the carpool problem maintains unfairness n against an adaptive adversary).

Fair queuing Consider a system where processes repeatedly access resources, each of which can be used by one process at a time. For simplicity assume that whenever a process uses a resource it takes one unit of time. We would like to find a fair way of ordering processes that try to access the same resource simultaneously. Given an execution in which a process tries to access a resource t times, with d_i processes at time i, its total expected waiting time, if the processes are scheduled at random, is $\sum_{i=1}^{t} \frac{d_i - 1}{2}$. We consider this as the process fair waiting time. Our goal is therefore to find a method of queuing the processes so as to minimize the difference between the actual time a process spends waiting for resources throughout its execution and its fair waiting time. We refer to this difference as the unfairness of the algorithm.

Note that we are interested in algorithms where the unfairness is a function of the number of processes, but is independent of the life-time of the system. Therefore, an algorithm that queues at random the processes that try to access the same resource simultaneously will not do, since with high probability there will be a process that is $\sqrt{t}$ away from its fair waiting time.

In the full version of the paper we relate the fair queuing problem to a weighted variant of the vector rounding problem and show that the maximum discrepancy between the fair waiting time and actual waiting time can be bounded by a function of the number of processes only.

Comparison with competitive analysis A popular methodology for evaluating the performance of on-line algorithm is the the *competitive analysis* approach of Sleator and Tarjan [21]: the on-line algorithm is compared with a "hypothetical" optimal off-line and bounds on the *competitive ratio* are obtained (for an adaptive, oblivious or random adversary). For the carpool problem, if one is given in advance a list specifying for each of the days which subset arrives on that day, then it is possible to construct a schedule whose unfairness is bounded by one (see Section 4). Therefore we can treat the results as being about the *competitive difference* of the carpool problem. If, instead, we would have analyzed the ratio between the fair-share = offline-load and the actual load, we would have obtained a $1+o(1)$ competitive ratio.

1.3 Background

Our problem is related to a "chip game" analyzed in [2]. In that game chips are placed in stacks on the integers, and in each round, two chips which are in the same stack may be selected, and one of them moved one step to the right while the other is moved one step to the left. This is the same thing that happens in the edge orientation game when a pair of vertices with the same indegree-outdegree difference is given; the difference between the games is that in ours pairs which are not colocated may also be selected (and moved toward each other). While our game can continue ad infinitum, the game of [2] must terminate, and some of the principal results of that paper concern the terminating states. In particular it is shown there that from any initial state of chips, there is a unique terminating position; and when n chips start all at the origin, no chip can be brought to distance more than $\lceil (n-1)/2 \rceil$ from the origin.

One can obtain an upper bound of $\lceil (n-1)/2 \rceil$ on the maximum unfairness of the greedy algorithm for the edge orientation problem by a reduction to the case of the game of [2]. The reduction is to show that for the greedy algorithm, any sequence of requests for pairs

of nodes can be replaced by another sequence, which reaches the same unfairness, but which uses no requests involving non-colocated pairs. (This reduction has also been noted recently by Babu Narayanan.) A more general proof is given in this paper.

1.4 Organization

Due to lack of space, we do not provide the details of many of the arguments; they are available in the full version of our paper [1]. In this extended abstract we describe in detail only the $O(\log n)$ analysis for the case of uniformly distributed requests on edges (Section 2), the general proof guaranteeing bounded unfairness (Section 3), and the reduction from the vector rounding problem to the edge orientation problem (Section 4). An overview of the $\Theta(\log\log n)$ analysis for the case of uniformly distributed requests on edges appears in Section 2.

2 Uniformly Distributed Requests

2.1 Overview

We present two analyses of the greedy algorithm for the edge orientation problem, for the case where the adversary schedules the edges uniformly at random. We say that a node is *at position* i if and only if its indegree$-$outdegree $= i$. In both methods the combination of the adversary and the algorithm is represented as a Markov process. A single state of the Markov chain is described by the list consisting of the number of nodes at each position.

When two nodes are paired, they either each move one step away from the other, if they are on the same point; or each move one step toward the other if they are not. Intuitively, we can think of the process as a balance between a "repulsive force" between colocated nodes and an "attractive force" between distant ones. To stretch this physical analogy further, we would expect that the attractive force, being stronger in spread-out configurations, would tend to gather the nodes into a tight clump held apart only by pressure from the repulsive force.

Our first analysis of the system, in Section 2.2, shows that the nodes do in fact clump together, and that in the stationary distribution of the Markov chain the expected maximal unfairness is $O(\log n)$. We define a potential function on the states of the system, in which each node contributes an amount that is exponential in its deviation from 0 in that state. Since the Markov chain corresponding to the system is ergodic when $n \geq 3$, we can use the fact that in the stationary distribution the expected change in the value of the potential function is 0. We show that at any state where the maximal unfairness exceeds $O(\log n)$, the potential function is likely to drop by a large amount: the expected change in the value of the potential function is at most $-n+1$. On the other hand we show that from any state, the potential function can rise by at most 1. For these small rises to balance out the large drops in the states with unfairness greater than $O(\log n)$, the probability of "high" unfairness can be at most $O(1/n)$; and since (as we show) the unfairness of any state cannot exceed n, the expected unfairness is just $O(\log n)$.

However, in our simulations of the process it appeared that the maximum unfairness of the global greedy algorithm against a uniform random adversary was much smaller than $O(\log n)$. Motivated by this observation, we examined the process more closely. The result was the tight asymptotic bounds of $\Theta(\log\log n)$ on the expected maximum unfairness. The details of these bounds are given in the full version of the paper [1]. Here, we limit ourselves to an overview of the method used to obtain them.

We obtain the $O(\log\log n)$ upper bound through a sequence of tighter and tighter approximations. To begin with, we pick a time interval of length $n^{\log\log n}$, whose starting point t is any point in the execution. We show that with high probability, the maximal unfairness goes below $\log n$ by time $t+n^4$, and then stays below $2\log n$ throughout the interval. We then remove the first n^4 steps in the interval from consideration. Next we show that, for each $\epsilon > 0$, with high probability we can chop off a prefix of this new interval whose length is polynomial in n, leaving a suffix in which the unfairness of all but ϵn nodes is bounded by a constant k. For any such interval we show that with high probability, we can chop off a second prefix, whose length depends polynomially on n but not at all on k or ϵ, to leave a suffix in which at most $\epsilon^2 n$ nodes are above $k+2$. Repeating this operation $\log\log n$ times gives us an interval whose length is only polynomially less than the interval we originally started with, and in which the maximal unfairness is at most $O(\log\log n)$ (with high probability). Since for sufficiently large n the low-unfairness interval is much longer than the high-unfairness interval, it dominates the average and thus gives an $O(\log\log n)$ upper bound on the expected unfairness.

This analysis is tight: by time $t+n^5$, the unfairness is at least $\log\log n$ and stays above $\log\log n$ for at least $n^{\log n}$ additional steps. The proof of this lower bound mirrors the proof of the upper bound. We show that with high probability, any sufficiently long interval throughout which the unfairness of at least ϵn nodes is at least k contains a suffix, whose starting point is polynomially shifted, in which the unfairness of at least $c\epsilon^2 n$ nodes is at least $k+1$; after $\log\log n$

iterations of this process we are left with an interval whose length is close to the length of the original interval we picked, such that with high probability, throughout this resulting interval, the maximal unfairness is at least $\log\log n$.

2.2 The $O(\log n)$ Bound

We maintain a *position* d_j for every $j \in [n]$. Initially, $d_j = 0$ for all $j \in [n]$. Given a request for a pair of nodes, the algorithm increases by one the position of the node whose current position is the smallest among the two, and decreases by one the position of the other particle in the pair. If two particles in the same position are requested, we flip an unbiased coin to determine which goes up and which goes down. Other positions remain the same. This is a randomized version of the *global greedy* strategy of [17]. We assume that the sequence generated is very long. The exact meaning of "very long" will be explained shortly. (We note that randomization of the on-line player is not essential to the analysis since, against the uniformly random adversary, the nodes may be considered unlabeled.)

Given such random input, the behavior of the global greedy algorithm can be represented as a Markov chain. By our analysis of the deterministic global greedy performance in the next section, we know that $|d_j| \leq \lceil (n-1)/2 \rceil$ for all $j \in [n]$. Thus, if the nodes are labeled, then the state space is $\{-\lceil (n-1)/2 \rceil, \ldots, \lceil (n-1)/2 \rceil\}^n$. The i-th coordinate of a state s, denoted s_i, is the position of the i-th node on the line. We now define the transitions and their probabilities. Let s be a state and $\{i,j\}$ a possible request. Without loss of generality, assume $s_i \leq s_j$. If $s_i < s_j$, then with probability $\binom{n}{2}^{-1}$ there is a transition to s' with $s'_i = s_i + 1$, $s'_j = s_j - 1$ and for all $k \notin \{i,j\}$, $s'_k = s_k$. If $s_i = s_j$, then with probability $\binom{n}{2}^{-1}/2$ there is a transition to s' as above, and with the same probability there is a transition to s'' with $s''_i = s_i - 1$, $s''_j = s_j + 1$ and for all $k \notin \{i,j\}$, $s''_k = s_k$. For $n \geq 3$ it is easy to see that limited to the set of states reachable from the initial state of the all-zero vector, this Markov chain is ergodic and therefore converges to a stationary distribution. We are interested in the long-term behavior of the chain and therefore assume that the adversary sequence is long enough for the stationary behavior to be dominant.

If the nodes are unlabeled, which we will assume in this section since we are considering the uniformly random adversary, then effectively we are interested in a smaller Markov chain, which is a coarsening of the above chain, and in which each state is simply a configuration of stacks of nodes lying on the integers: thus, the state is represented by a vector $n_{-\lceil (n-1)/2 \rceil}, \ldots, n_{\lceil (n-1)/2 \rceil}$ where each n_i is the number of particles at position i. Let $p_j = n_j/n$. Note that if s is a state reachable from the all 0's vector, then $\sum_i i n_i = 0$.

Let $\alpha = \frac{3}{2}$ and let the potential function

$$\Phi(s) = \sum_{j=-\lceil (n-1)/2 \rceil}^{\lceil (n-1)/2 \rceil} n_j \cdot \alpha^{|j|}.$$

Let $\Delta\Phi(s) = E_{s'}[\Phi(s')] - \Phi(s)$, where s' denotes the (random) state reached from s in one step of the Markov chain.

We wish to estimate $\Delta\Phi(s)$. The following fact is easily verified:

FACT 2.1. *If s' is any outcome of requesting two nodes occupying the same position, then $\Phi(s') - \Phi(s) > 0$. If s' is the outcome of requesting two nodes that are at distance 1 apart, then $\Phi(s') - \Phi(s) = 0$. Otherwise, $\Phi(s') - \Phi(s) < 0$.*

Estimating $\Delta\Phi(s)$ is done by estimating the contribution of each position separately, and adding up those contributions. The idea is to show that for any $j \neq 0$, the positive contribution due to two nodes in j being requested is overwhelmed by the negative contribution due to a node in j and a node on the other side of 0 being requested. We will ignore other requests. They can only increase the negative contribution. In order to do this correctly, we need to consider disjoint events, so, to evaluate the contribution of position j, we will consider ordered pairs, where the first of the two is from j. The following fact is also easily verified:

FACT 2.2. *Under a uniform distribution over pairs of nodes,*

$$Prob[j,j] = Prob[\text{ordered } j,j] \leq p_j^2,$$

and

$$\begin{aligned} \frac{1}{2} Prob[i,j,i \neq j] &= Prob[\text{ordered } i,j] \\ &= Prob[\text{ordered } j,i] \geq p_i p_j. \end{aligned}$$

These relations are inequalities (rather than equalities) because we draw each pair without replacement; this makes it slightly less likely that we will draw two nodes at the same location.

Let A_j be the event that the first node in a pair is j (given that we are at configuration s). Formally, the *contribution* of position j to $\Delta\Phi(s)$ is $p_j E[\Phi(s') - \Phi(s) | A_j]$. We now show:

LEMMA 2.1. *For j, $1 \leq |j| \leq \lceil (n-1)/2 \rceil$, the contribution of position j to $\Delta\Phi(s)$ is at most $-\frac{1}{6} p_j^2 \alpha^{|j|}$.*

Proof. Let $j > 0$. The argument for $j < 0$ is symmetric. If the pair j,j is chosen, the increase in the potential function is $\alpha^{j+1} + \alpha^{j-1} - 2\alpha^j$. On the other hand, if the (ordered) pair chosen is j,i ($i < 0$), the

decrease in the potential function is $\alpha^j+\alpha^i-\alpha^{j-1}-\alpha^{i+1}$. We need an estimate on the distribution of nodes on the negative side. Since the sum of the positions of the nodes is 0, the n_j nodes at j must be balanced by nodes in negative positions. Hence:

$$\sum_{i<0}(-i)n_i \geq j \cdot n_j .$$

It is not difficult to see that the worst case (the least decrease in Φ) is when equality holds and when all the negative side nodes are in one position $-x$. Notice that we might need to consider a non-integral x. So, we get $xn_{-x} \geq jn_j$, or $p_{-x} \geq \frac{j}{x}p_j$. The total decrease in the potential function due to position j is at least:

$$p_j^2(\quad \tfrac{j}{x}\left(\alpha^j+\alpha^x-\alpha^{j-1}-\alpha^{x-1}\right) \\ -\alpha^{j+1}-\alpha^{j-1}+2\alpha^j),$$

for x minimizing this expression. Observe that this decrease is "essentially" the sum of two first derivatives of α^j, minus its second derivative. The basis of our lower bound on this expression is that for α below some threshold, the increase due to the first derivative (representing the choice of nodes at two different locations) dominates the decrease due to the second derivative (representing the choice of colocated nodes).

We show that for $j > 0$ the decrease is at least $p_j^2\alpha^j/6$, i.e. that for all $j \geq 1$ and $x > 0$,

$$\begin{aligned} & j\left(\alpha^j+\alpha^x-\alpha^{j-1}-\alpha^{x-1}\right) \\ > \quad & x\left(\alpha^{j+1}+\alpha^{j-1}-2\alpha^j+\frac{1}{6}\alpha^j\right), \end{aligned}$$

or

$$j\left(1-\alpha^{-1}\right)\left(\alpha^j+\alpha^x\right) > x\left(\alpha+\alpha^{-1}-\frac{11}{6}\right)\alpha^j .$$

Recalling that $\alpha = \frac{3}{2}$ we want to prove that

$$\frac{1}{3}j\left(\alpha^j+\alpha^x\right) > \frac{1}{3}x\alpha^j .$$

If $x \leq j$ this is trivial. If $x > j$ write $r = x - j > 0$. We wish to show that $j\alpha^r > r$, and since $j \geq 1$ it suffices to show that $\alpha^r > r$ for all $r > 0$. Let $\delta = \min_r \alpha^r - r$. Some calculus shows that δ is achieved at $r = \frac{-\log\log\alpha}{\log\alpha}$; and moreover that δ varies monotonically in α. Thus we can solve for $\delta = 0$, finding that this is achieved for $\alpha = e^{1/e} \approx 1.4447$, and conclude that $\delta > 0$ for all $\alpha > e^{1/e}$ and in particular for the chosen $\alpha = \frac{3}{2}$. ■

For $j = 0$ we cannot guarantee a negative contribution. However, we can upper bound the conditional positive contribution by $2\alpha - 2 = 1$, since the probability of choosing a pair in positions $0, 0$ is at most 1 and the total increase due to these positions is at most 1.

Concluding the above discussion: the contribution of position 0 is at most $+1$. The contribution of position j, $|j| \geq 1$ is at most $-\frac{1}{6}p_j^2(\frac{3}{2})^{|j|}$.

Let $T = 3\log_{\frac{3}{2}} n + \log_{\frac{3}{2}} 6$ and assume $n \geq 3$. Then, if s has a node whose distance from 0 is more than T, then $\Delta\Phi(s) \leq -n+1$ (note that if this node's position is j, then $p_j \geq \frac{1}{n}$).

Now, partition the state space into two subsets: A contains those states that do not contain a node beyond T; B contains the other states. We have

FACT 2.3. *$\forall a \in A$, $\Delta\Phi(a) \leq 1$. $\forall b \in B$, $\Delta\Phi(b) \leq -n+1$.*

Since the total expected change in Φ, under the stationary distribution, must be 0, it must hold that under the stationary distribution, $Prob[B] \leq \frac{1}{n}$. It follows that:

THEOREM 2.1. *For $n \geq 3$, in the stationary distribution, the probability that any node is beyond distance $T = 3\log_{\frac{3}{2}} n + \log_{\frac{3}{2}} 6$ from the origin is at most $1/n$. Thus the expectation of the maximum distance of a node from the origin is $\leq T + 1 = O(\log n)$.*

3 Bounded Unfairness

We consider the following on-line deterministic strategy for the n-participant carpool game. We maintain the *deviation* d_j for every $j \in [n]$. Initially, $d_j = 0$ for all $j \in [n]$. Given a request r (i.e., a subset of $[n]$ of cardinality 2 or more), the algorithm chooses $j \in r$ such that $d_j = \min_{i \in r} d_i$, breaking ties arbitrarily. The deviations are then updated as follows. d_j increases by $1 - 1/|r|$. For all other elements $i \in r$, $i \neq j$, d_i decreases by $1/|r|$. Other deviations remain the same. This strategy is the *global greedy* strategy of [17].

We show an upper bound on the unfairness resulting from the deterministic global greedy algorithm. The following fact is trivial.

FACT 3.1. *For any adversary ϱ, the unfairness of global greedy is given by $\max_{j\in[n]}|d_j|$, where the values of d_j are taken at the end of the game.*

LEMMA 3.1. *Consider an n-participant carpool game between any adversary and the global greedy algorithm. For every round of the game there exists a weighted directed graph with node set $[n]$, edge set E and weight function w with the following properties.*

1. $$\forall e \in E,\ \frac{1}{n!} \leq w(e) \leq \frac{1}{2}.$$

2. *$\forall e \in E$, $w(e) = \frac{p}{q}$, where p, q are integers, and q divides $n!$.*

3.

$$\forall j \in [n],\ d_j = \sum_{e \in in(j)} w(e) - \sum_{e \in out(j)} w(e)$$

where $in(j)$ is the set of incoming edges incident to j and $out(j)$ is the set of outgoing edges incident to j.

Proof. The proof is by induction on the number of rounds.

Basis: No rounds. Take an empty graph.

Inductive step: Assume the claim holds for $t-1$ rounds. Let the t-th request of the adversary be $X_t = \{i_1, \ldots, i_k\}$.

Define $w(i,j)$ to be the weight of the directed edge from i to j, if such exists, or minus the weight of the directed edge from j to i, if such exists, or 0 otherwise.

Without loss of generality, assume that the global greedy algorithm selects $i = i_1$. We will modify the graph in two steps. The first modification, described below, maintains the two conditions of the lemma for round $t-1$ and in addition establishes that there is no edge to i from any other node in X_t. The second step adds an edge with weight $1/k$ from every i_j, $2 \le j \le k$ to i. If these new edges create any pairs of anti-parallel edges, we merge each such pair to a single directed edge. Its weight is the difference between the larger and the smaller weight in the pair, and its direction coincides with the larger weighted edge in the pair. (If the two weights are equal, we remove both edges from the graph.)

To complete the proof we show how to do the first step. Suppose there exists $j \in X_t$ such that $w(j,i) > 0$. From the definition of the algorithm we have that after round $t-1$, $d_i \le d_j$. Thus, there exists $l \in [n]$ such that $w(l,j) > w(l,i)$.

We execute the following procedure.

while $w(j,i) > 0$ **do**
  Choose $l \in [n]$ such that $w(l,j) > w(l,i)$
  if $w(l,j) > 0$ **then**
    Let $w = \min\{1/2 - w(l,i), w(j,i), w(l,j)\}$,
    Increase $w(l,i)$ by w,
    Decrease $w(l,j)$ and $w(j,i)$ by w each.
  else $(w(l,j) < 0)$
    Let $w = \min\{1/2 - w(j,l), w(j,i), w(i,l)\}$,
    Increase $w(j,l)$ by w,
    Decrease $w(j,i)$ and $w(i,l)$ by w each.
stop .

Recall that by the inductive hypothesis, at the beginning of round t, $w(e) = \frac{p}{q}$, where p, q are integers and q divides $n!$. Clearly, this property is preserved by the above procedure (note that $n \ge 2$). Moreover, it implies that $w(e)$ continues to be $\ge 1/n!$ unless it becomes 0. Thus in both cases of the procedure above, in each iteration the sum of the weights over all edges in the graph decreases by $w \ge 1/n!$. Therefore, this process terminates. ■

We conclude from Fact 3.1 and Lemma 3.1 that

THEOREM 3.1. *For any adversary ϱ, the unfairness of global greedy is at most $\lceil \frac{n-1}{2} \rceil$.*

In comparison, we note that the results of [2] provide a lower bound on the performance of any deterministic algorithm. Stated in our framework, it is shown there that in the edge orientation problem, any sequence of requests which picks only colocated pairs of nodes, will eventually result in the nodes occupying the entire interval $[-\lceil (n-1)/2 \rceil, \lceil (n-1)/2 \rceil]$ (except the origin in case n is even). Moreover it is shown there that the number of requests necessary to bring the nodes to this configuration is $n(n+1)(n+2)/24$ if n is even, and $(n-1)n(n+1)/24$ if n is odd. We use this bound on the length of the request sequence later on to prove lower bounds for randomized algorithms.

Thus:

THEOREM 3.2. ([2]) *For every deterministic edge orientation algorithm f, for every $k \in \boldsymbol{Z}^+$, $k \le \lceil \frac{n-1}{2} \rceil$, there exists an adversary ϱ that gives a sequence of at most k^3 requests, pushing the unfairness achieved by f to at least k.*

(To apply the bounds to the carpool problem, divide the unfairness by 2.)

4 Reducing Vector Rounding to the 2-Person Carpool Game

In this section we show that the general carpool problem can be reduced to one where each day only two people arrive. This is done by a reduction from the still more general *vector rounding problem*. The n-dimensional vector rounding problem is this: the input is a list of vectors $(V_1, V_2, \ldots)$, where each $V_t = (v_t^1, v_t^2, \ldots v_t^n)$ is a vector of length n over the reals. The output is a list of integer vector $(Z_1, Z_2, \ldots)$ where $Z_t = (z_t^1, z_t^2, \ldots z_t^n)$ is a rounding of V_t that preserves the sum, i.e. for all $1 \le i \le n$ we have that $z_t^i \in \{\lfloor v_t^i \rfloor, \lceil v_t^i \rceil\}$ and that

$$\sum_{i=1}^{n} z_t^i \in \{\lfloor \sum_{i=1}^{n} v_t^i \rfloor, \lceil \sum_{i=1}^{n} v_t^i \rceil\}$$

The goal is to make the accumulated difference in each entry as small as possible, i.e. for every t we want $\max_{1 \le i \le n} |\sum_{j=1}^{t} z_j^i - \sum_{j=1}^{t} v_j^i|$ to be as small as possible. For input vectors $(V_1, V_2, \ldots)$ and output vector $(Z_1, Z_2, \ldots)$ the associated cost at time t is $\max_{1 \le i \le n} |\sum_{j=1}^{t} z_j^i - \sum_{j=1}^{t} v_j^i|$. As before we can consider the off-line problem where we are given the vectors $(V_1, V_2, \ldots)$ ahead of time and the on-line problem where we are given the vectors $(V_1, V_2, \ldots)$ one at a time and have to decide on the corresponding $(Z_1, Z_2, \ldots)$.

As in the carpool problem, in the on-line version we consider deterministic algorithms as well as randomized algorithms against the oblivious adversary.

Tijdeman [23] has considered the vector rounding problem and has shown that the off-line version has a solution of difference 1, i.e. for every sequence of real vectors $(V_1, V_2, \ldots)$ there exist integer $(Z_1, Z_2, \ldots)$ such that for all $t \geq 1$ we have $\max_{1 \leq i \leq n} |\sum_{j=1}^{t} z_j^i - \sum_{j=1}^{t} v_j^i| \leq 1$.

One can easily cast the carpool problem as a vector rounding problem: for a sequence $(X_1, X_2, \ldots)$ create the vectors $(V_1, V_2, \ldots)$ where for all $t \geq 1$ and all $1 \leq i \leq n$ we have

$$v_t^i = \begin{cases} 1/|X_t| & \text{if } i \in X_t \\ 0 & \text{if } i \notin X_t. \end{cases}$$

Therefore if we can connect the performance of the 2-person carpool problem to the vector rounding problem then we will have reduced the general carpool problem to the 2-person problem.

Before we show the reduction we will make some simplifying assumptions, which can be easily justified: We assume that for every t and $1 \leq i \leq n$, v_t^i is non-negative (since we can add the absolute value of $\lceil v_t^i \rceil$ and then subtract it from z_t^i) and that $\sum_{i=1}^{n} v_t^i$ is an integer (if not, it only gives us more freedom). Furthermore we assume an a priori bound T on the number of vectors, i.e. $t < T$ (otherwise we will increase T as we go along in multiples of 2).

Our reduction is applicable to both deterministic and randomized algorithms.

THEOREM 4.1. • *Deterministic Algorithms: Suppose that we have a deterministic algorithm f for the n-participant carpool problem where every day two people show up that maintains unfairness at most $\alpha(n)$, then we can construct a deterministic algorithm f' to the vector rounding problem that maintains an accumulated difference of at most $2\alpha(n)$ for every sequence $\varphi = (V_1, V_2, \ldots)$.*

- *Randomized algorithms against the oblivious adversary: Suppose that we have a randomized algorithm $\tilde{f}$ for the n-participant carpool problem where every day two people show up that maintains unfairness $\alpha(n)$, then we can construct a randomized algorithm $\tilde{f}'$ for the vector rounding problem that maintains an accumulated difference of at most $2\alpha(n)$, for every sequence $\varphi = (V_1, V_2, \ldots)$.*

Proof. The reduction is made by a "scaling" argument, similar in flavor to the bit-by-bit rounding of Beck and Fiala [10, 9]. The constructions of the deterministic and randomized algorithms for the vector rounding problem from the corresponding algorithms for the 2-person carpool problem are similar, only the analysis is a bit different. Consider the binary representation of the v_t^i's. We will make it only $\ell = 2 \log T$ bits long by ignoring the rest of the bits and adjusting one of the v_t^i. This can hardly affect the outcome (by a $\frac{1}{T}$ additive term only, and this can be made arbitrarily small by making ℓ larger). Run ℓ carpool instances in parallel, one corresponding to each of the ℓ bit positions. For each instance apply the strategy for the carpool problem (f or $\tilde{f}$ depending on the case). Each problem has n participants and the accounting and decisions (But not the inputs!) of each instance are done independently.

We start by describing how the ℓ-th instance is defined and then how the rest of the instances follow. Consider the ℓ-th bits of the entries of V_t. Since $\sum_i v_t^i \in \boldsymbol{Z}^+$, there must be an even number of i's such that the ℓ-th bit of v_t^i is 1. Partition them into pairs arbitrarily and schedule those pairs as requests. If v_t^i and v_t^j are paired, request $\{i, j\}$. Those i's that were chosen to drive by the carpool strategy f or $\tilde{f}$ are rounded up, i.e. we add 2^{-l} to v_t^i. Those i's that were not chosen as drivers are rounded down, i.e. we simply throw away (that is, replace with a 0) the ℓ-th bit of v_t^i. It is easy to verify that this procedure preserves the sum of entries (i.e. $\sum_{i=1}^{n} v_t^i$) and that the modified V_t requires only $\ell - 1$ bits for its representation. The procedure is repeated now with the $\ell - 1$st instance and so on. After we do that for all the ℓ carpool instances we are left with an integer V_t which is our Z_t.

How good is this reduction? Let $C(t, j, i, V_1, V_2, \ldots V_t)$ denote the unfairness of the i-th participant at the j-th instance defined by inputs $V_1, V_2, \ldots V_t$. Let $D(t, i, V_1, V_2, \ldots V_t)$ be $\sum_{j=1}^{t}(z_j^i - v_j^i)$ on input $V_1, V_2, \ldots V_t$. We claim that

$$\begin{aligned} & D(t, i, V_1, V_2, \ldots V_t) \\ = & \sum_{j=1}^{\ell} \frac{1}{2^{j-1}} C(t, j, i, V_1, V_2, \ldots V_t) \end{aligned}$$

This can be shown by induction on ℓ. The contribution of the ℓ-th instance is $1/2^{\ell-1}$, since we have scaled the ℓ-th instance by $2^{\ell-1}$.

We now turn to the analysis. In the deterministic case we know by assumption on f that $|C(t, j, i, V_1, V_2, \ldots V_t)|$ is bounded by $\alpha(n)$. Therefore,

$$\begin{aligned} & |D(t, i, V_1, V_2, \ldots V_t)| \\ \leq & \sum_{j=1}^{\ell} \frac{1}{2^{j-1}} |C(t, j, i, V_1, V_2, \ldots V_t)| \leq 2\alpha(n) \end{aligned}$$

and therefore $\max_{1 \leq i \leq n} |D(t, i, V_1, V_2, \ldots V_t)| \leq 2\alpha(n)$

For the case of a randomized algorithm, we should first be convinced that the adversary's power is no stronger than that of an oblivious adversary in each of the carpool instances that we have defined. Observe that the inputs to the j instance are determined by $(V_1, V_2, \ldots)$ and the decisions made by the carpool solver on instances $j+1$ through ℓ. The decisions made in instance k, for $1 \leq k \leq j$ at *any* point in time do not effect the inputs to instance j. Therefore, for instance j, the adversary chooses a distribution on $(V_1, V_2, \ldots)$ and can even be given the power to make all the decisions in instances $j+1$ through ℓ and yet all that it would be doing cannot depend on the decision at the j-th instance. Given that we know that

$$\begin{aligned} & \max_{1 \leq i \leq n} |D(t, i, V_1, V_2, \ldots V_t)| \\ \leq & \sum_{j=1}^{\ell} \frac{1}{2^{j-1}} \max_{1 \leq k \leq n} |C(t, j, k, V_1, V_2, \ldots V_t)| \end{aligned}$$

and the expectation of

$$\max_{1 \leq k \leq n} |C(t, j, k, V_1, V_2, \ldots V_t)|$$

is bounded by $\alpha(n)$, we get that the expectation of $\max_{1 \leq i \leq n} |D(t, i, V_1, V_2, \ldots V_t)|$ is at most $2\alpha(n)$. ■

Acknowledgments

We are grateful to Noga Alon, Tomás Feder, Alan Frieze, Anna Karlin, Nimrod Megiddo, Babu Narayanan, Gerald Schedler and Joel Spencer for many helpful discussions.

References

[1] M. Ajtai, J. Aspnes, M. Naor, Y. Rabani, L. J. Schulman and O. Waarts. Fairness in Scheduling manuscript, available from the authors.

[2] R. Anderson, L. Lovász, P. Shor, J. Spencer, E. Tardos and S. Winograd. Disks, balls, and walls: analysis of a combinatorial game. American Mathematical Monthly **96**, 1989, pp. 481–493.

[3] T. E. Anderson, S. Owicki, J. B. Saxe and C. P. Thacker. High Speed Switch Scheduling for Local Area Networks. *ACM Trans. on Comm. Syst.*, 11(4), 1993, pp. 319–352.

[4] J. Aspnes, Y. Azar, A. Fiat, S. A. Plotkin, and O. Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. In *Proc. of the 25th Ann. ACM Symp. on Theory of Computing*, pages 623-631, May 1993.

[5] Y. Azar, A. Broder, and A. Karlin, E. Upfal. Balanced allocations. In *Proc. of the 26th Ann. ACM Symp. on Theory of Computing*, pages 593–602, May 1994.

[6] Y. Azar, J. Naor, and R. Rom. The competitiveness of on-line assignment. In *Proc. of the 3rd Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 203–210, 1992.

[7] Zs. Baranyai. On the factorization of the complete uniform hypergraph. In *Infinite and finite sets*, Colloquia Mathematica Societatis János Bolyai, 1973.

[8] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. In *Proc. of the 24th Ann. ACM Symp. on Theory of Computing*, pages 51–58, 1992.

[9] J. Beck. Balanced two-colorings of finite sets in the cube. *Discrete Math*, 73:13–25, 1988–9.

[10] J. Beck and T. Fiala Integer-making theorems. *Discrete Applied Mathematics*, 3:1–8, 1981.

[11] S. Ben-David, A. Borodin, R.M. Karp, G. Tardos, and A. Wigderson. On the power of randomization in online algorithms. In *Proc. of the 22nd Ann. ACM Symp. on Theory of Computing*, pages 379–386, May 1990.

[12] A. Borodin, N. Linial, and M. Saks. An optimal on-line algorithm for metrical task systems. In *Proc. of the 19th Ann. ACM Symp on Theory of Computing*, pages 373–382, May 1987.

[13] A. Charny. An Algorithm for Rate Allocation in a Packet-Switching Network With Feedback MIT/LCS TR-601, 1994.

[14] D. Coppersmith. Private Communication.

[15] E. Davis and J.M. Jaffe. Algorithms for scheduling tasks on unrelated processors. *JACM*, 28:712–736, 1981.

[16] A. Demers, S. Keshav and S. Shenkar Analysis and simulation of a fair queuing algorithm Proc. ACM SIGCOMM 89, pp. 1–12.

[17] R. Fagin and J.H. Williams. A fair carpool scheduling algorithm. In *IBM Journal of Research and Development*, 27(2):133-139, March 1983.

[18] R.L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[19] E. Hahne. Round-robin scheduling for maxmin fairness in data networks *IEEE J. on Selected areas in Communication, vol 9(7), 1991.*

[20] K. Ramakrishnan and R. Jain. A binary feedback scheme for congestion avoidance in computer networks. *ACM Trans. on Comm. Syst.*, 8(2), 1990, pp. 158–181.

[21] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communication of the ACM*, 28(2):202–208, 1985.

[22] D. Shmoys, J. Wein, and D. P. Williamson. Scheduling parallel machines on-line. In *Proc. of the 32nd IEEE Ann. Symp. on Foundations of Computer Science*, pages 131–140, 1991.

[23] R. Tijdeman. The chairman assignment problem. *Discerte Math*, 32:323–330, 1980.

[24] A.C. Yao. Probabilistic computation, towards a unified measure of complexity. In *Proc. of the 18th Ann. IEEE Symp. on Foundation of Computer Science*, pages 222–227, 1977.