Carpooling in Social Networks

Amos Fiat*[1], Anna R. Karlin[2], Elias Koutsoupias[3], Claire Mathieu[4], and Rotem Zach[1]

1 **Department of Computer Science**
Tel Aviv University
2 **Department of Computer Science**
University of Washington in Seattle
3 **Department of Computer Science**
University of Oxford
4 **Department of Computer Science**
CNRS, Ecole Normale Superieure

"It was then that Hook bit him. Not the pain of this but its unfairness was what dazed Peter."

— J.M. Barrie, *Peter Pan* (1911)

Abstract

We consider the online carpool fairness problem of [Fagin and Williams, 1983] in which an online algorithm is presented with a sequence of pairs drawn from a group of n potential drivers. The online algorithm must select one driver from each pair, with the objective of partitioning the driving burden as fairly as possible for all drivers. The *unfairness* of an online algorithm is a measure of the worst-case deviation between the number of times a person has driven and the number of times they would have driven if life was completely fair.

We introduce a version of the problem in which drivers only carpool with their neighbors in a given social network graph; this is a generalization of the original problem, which corresponds to the social network of the complete graph. We show that for graphs of degree d, the unfairness of deterministic algorithms against adversarial sequences is exactly $d/2$.

For random sequences of edges from planar graph social networks we give a [deterministic] algorithm with logarithmic unfairness (holds more generally for any bounded-genus graph). This does not follow from previous random sequence results in the original model, as we show that restricting the random sequences to sparse social network graphs may increase the unfairness.

A very natural class of randomized online algorithms are so-called static algorithms that preserve the same state distribution over time. Surprisingly, we show that any such algorithm has unfairness $\tilde{\Theta}(\sqrt{d})$ against oblivious adversaries. This shows that the *local random greedy* algorithm of [Ajtai et al, 1996] is close to optimal amongst the class of static algorithms. A natural (non-static) algorithm is global random greedy (which acts greedily and breaks ties at random). We improve the lower bound on the competitive ratio from $\Omega(\log^{1/3}(d))$ to $\Omega(\log d)$. We also show that the competitive ratio of global random greedy against adaptive adversaries is $\Omega(d)$.

1998 ACM Subject Classification F.1.2 Online Computation, F.2.0 Analysis of Algorithms and Problem Complexity

Keywords and phrases Online algorithms, Fairness, Randomized algorithms, Competitive ratio, Carpool problem

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.XXX

* Both Amos Fiat and Rotem Zach were supported by Israel Science Foundatation grant no. 1841-14

1 Introduction

In multiple experimental studies involving hundreds of graduate students, Loewenstein, Thompson, and Bazerman [8] give evidence that individuals are strongly averse to outcomes where they are at a disadvantage relative to others. Moreover, albeit significantly less so, the grad students were also averse to outcomes where they have a relative advantage in payoff. Fehr and Schmidt [6] coined the phrase *inequity aversion* to describe this phenomena. Festinger [7] had much earlier introduced the concept of cognitive dissonance, and inequity aversion is modeled as a special case thereof. Supposedly, inequity aversion may lead individuals to make significant changes, including stopping interpersonal relationships where inequities arise.

The *carpool* problem, introduced by Fagin and Williams [5], is a stylized mathematical model in which one can study questions related to minimizing inequity. As described in [5], *"suppose that n people, tired of spending their time and money in gasoline lines, decide to form a carpool. We present a scheduling algorithm for determining which person should drive on any given day. We want a scheduling algorithm that will be perceived as fair by all the members."* A priori, it seems that fairness should not be hard to achieve, but — unfortunately — precise answers as to what extent one can avoid inequity have been sought over two decades with seemingly little progress.

Formally, each day t, a set of people $S_t \subseteq \{1, \ldots, n\}$ form a carpool. The goal is to choose who drives, so that on all days t, the overall driving burden to date has been partitioned fairly: Let $f_i(t)$ be driver i's fair share of the driving on day t, which is $1/|S_t|$ for each $i \in S_t$ and 0 otherwise. Define $F_i(t)$ to be driver i's fair share of the driving on all days up to day t, that is $F_i(t) = \sum_{\tau \le t} f_i(\tau)$, and let $D_i(t)$ be the number of times i has actually driven out of the first t days. For a particular sequence $\{S_t\}_{t=1}^{T}$, and algorithm for deciding who drives, we define

$$\text{the unfairness on day } t \;=\; \max_{\text{driver } i} |D_i(t) - F_i(t)|. \tag{1}$$

A carpool algorithm decides which person in S_t drives on day t; the *maximum unfairness* of the algorithm is

$$\max_{T \ge 1} \max_{\{S_t\}_{t=1}^{T}} [\text{unfairness on day } T]\,.$$

Notice that the definition of unfairness takes into account **all** trips i took, regardless of who i's companions were on that trip, that is, it is a global notion of fairness.

The offline version of the problem, when $\{S_t\}_{t=1}^{T}$ is known in advance, is easy: there is an algorithm that guarantees maximum unfairness of 1 and this is optimal (see, e.g. [10].)

Ajtai, Aspnes, Naor, Rabani, Schulman, and Waarts [1] studied the online version of problem, in which the algorithm must select a driver on day t, based only on the history up to time t. They obtained a number of extremely interesting results. First, they showed that, up to losing a factor of 2, one may assume that all the sets S_t consist of two persons. Thus, one can think of the process as a sequence of edge additions (*requests*), say $S_t = (i, j)$ at time t, to a multigraph on $\{1, \ldots, n\}$ (the people), with the algorithmic decision being one of choosing the orientation of the edge (towards the driver for that carpool). The goal then is to minimize

$$\max_{\text{vertex } i} \,|\, \text{indegree}(i) - \text{outdegree}(i) \,|\,.$$

Ajtai et al. obtained results for two different online settings: when the requests (carpools) are selected uniformly at random, and when the request sequence is selected by an

oblivious adversary that knows the algorithm, but not the outcome of any random choices the algorithm makes.

The first algorithm they considered was *Global Greedy*: on request (i, j), the driver among i and j with minimum unfairness (as defined in equation (1)) drives,; in case of a tie, the choice is arbitrary. For a uniformly random request sequence, they showed that for each t, *Global Greedy* has expected unfairness on that day of $O(\log \log n)$.

For the adversarial case, Ajtai et al showed that every deterministic algorithm has unfairness $\lfloor n/2 \rfloor$. They also showed that this is tight: *Global Greedy* has unfairness at most $n/2$ for every request sequence. They were able to obtain a better upper bound using *Randomized Local Greedy*: This algorithm considers each pair of drivers separately, and alternates which one drives each time they form a carpool. The only randomness is in the uniformly random choice of which of the two drives the very first time they carpool. They showed that *Randomized Local Greedy* has maximum unfairness equal to $\Theta(\sqrt{n \log n})$.

They conjectured that *Randomized Global Greedy*, the variant of *Global Greedy* in which ties are broken at random is much better, perhaps even polylog(n).

Finally, they proved that every randomized algorithm has maximum unfairness of at least $\Omega\left((\log n)^{1/3}\right)$.

Our Results

We study the carpool problem in the setting where the people involved belong to a social network G, and every request (carpool) is a pair of people that are connected in the social network, *i.e.* an edge of G. In this context, the work of [5, 1] can be seen as studying the special case where the social network is a clique.

We prove the following results for request sequences restricted to edges of a social network G with n vertices, and of maximum degree d.

Deterministic algorithms, adversarial requests

We show that for every deterministic algorithm there exists a request sequence on G resulting in unfairness of at least $\lfloor d/2 \rfloor$. This is tight: we give a deterministic algorithm that never generates unfairness greater than $d/2$ (Theorem 2.1).

What is most interesting about this result is that, in contrast to the case where the graph is complete, the optimal deterministic algorithm is *not* the *Global Greedy* algorithm. In fact, we show that for every connected G (irrespective of its maximum degree), there is a request sequence on which *Global Greedy* has worst-case unfairness $\geq \lfloor n/2 \rfloor$ (Theorem 2.2). Thus, *Global Greedy* can be a factor $\Omega(n)$ worse than the optimal deterministic algorithm (e.g., when the graph has constant degree).

Random requests

Our second set of results concerns random requests: We show that if the sequence of requests is generated by choosing edges of G uniformly at random, then the removal of edges from the graph can increase the unfairness for the *Global Greedy* algorithm: When G is a path, *Global Greedy* has expected unfairness at least $\Omega((\log n / \log \log n)^{1/3})$ (Theorem 3.3). This stands in contrast to the $O(\log \log n)$ upper bound of Ajtai et al when the graph G is a clique.

For a social network G of bounded genus (e.g., planar graphs, the torus, etc.) — we introduce the "star algorithm" - an algorithm with expected maximum unfairness $O(\log n)$ (Corollary 3.2).

Randomized algorithms, adversarial requests

Oblivious Adversaries:

Oblivious adversaries determine the event sequence in advance, the algorithm may toss coins, and one considers the expected cost to the algorithm.

The results of Ajtai et al. show that *Randomized Local Greedy* gives maximum expected unfairness of $O(\sqrt{d \log d})$, since each vertex has degree at most d and the unfairness at each node is the expected value of the sum of at most d random variables that are equally likely to be 1 or -1. One can view this algorithm as maintaining an invariant probability distribution over unfairness configurations: for each t, regardless of the history of requests, each edge is oriented uniformly at random. In this sense, it is a *static algorithm*. Static algorithms form a very natural class of randomized online algorithms. Intuitively, they render an adversary powerless to construct a bad request sequence: every request sequence will perform the same against such an algorithm. We show that unfortunately, this intuition is incorrect and that the competitive ratio of *every* static algorithm is at least $\Omega(\sqrt{d})$.

As mentioned above, the *Randomized Global Greedy* algorithm has been conjectured to give a good competitive ratio against oblivious adversaries. We prove that *Randomized Global Greedy* has unfairness $\Omega(\log n)$ (on a clique of size n), improving upon the previous lower bound of $\Omega\left((\log n)^{1/3}\right)$ from [1]. This involves a rather complex proof, for which we only give a high level sketch here, the full details are available online.

Adaptive Adversaries:

Adaptive adversaries determine the next event in the event sequence as a function of the previous responses of the online algorithm. *I.e.*, as a function of how the previous car pooling events were addressed.

We show that *no* randomized algorithm (static or not) has unfairness better than $d/4$ against an adaptive adversary [11].

Other Related Work

Another problem that can model fairness issues is Tijdeman's *chairman assignment problem* [10] where a chairman has to be appointed by a community of unequal groups. An axiomatic approach to the problem and its relationship to the Shapley value of a game was given in [9]. Generalizations of the carpool problem appear in [4, 3, 2].

Notation

In what follows, we often suppress the dependence on t in our notation for the unfairness of driver i at time t. Specifically, we use x_i to denote the unfairness of driver i at time t (i.e. $x_i := 2\big(D_i(t) - F_i(t)\big) = \text{indegree}(i) - \text{outdegree}(i)$), where t is understood. Note that $\sum_i x_i = \sum_i [\text{indegree}(i) - \text{outdegree}(i)] = 0$ at all times, and that we are assuming that all carpools are of size 2.

2 Deterministic algorithms

▶ **Theorem 2.1.** *Let G be a graph of maximum degree d. Then for every deterministic algorithm A, there exists a request sequence σ such that after A processes σ the unfairness is $\lceil d/2 \rceil$. This is tight: there is a deterministic algorithm such that after it processes every request sequence the unfairness is at most $\lceil d/2 \rceil$.*

Proof. For the first part, we restrict our sequences to the subgraph of G consisting of a maximum degree vertex and all of its neighbors, *i.e.,* a star with d leaves. For each deterministic algorithm, we will prove that either the unfairness is unbounded or there is a sequence of requests for which the root of the star has unfairness $\pm\lceil d/2\rceil$.

Fix a deterministic algorithm. We will say that it is in state $\vec{x} = (x_1, \ldots, x_d)$ when the unfairness of leaf i is x_i (any ordering of the children is fine) and therefore the unfairness of the root is $-\sum_1^d x_i$. Let S be the set of states that can be reached by some request sequence. We can assume that for all i, $|x_i| \leq \lceil d/2\rceil$; if not, we are done. Therefore, S is bounded.

Select $\vec{x} \in S$ which minimizes $\varphi(\vec{x}) = x_1 + 2x_2 + \cdots + 2^{d-1}x_d$. Let r be a request sequence that reaches $\vec{x}$. If we extend r with request $(root, 1)$, then by the minimality of $\varphi(\vec{x})$, the online algorithm is not allowed to move to $(x_1 - 1, \vec{x}_{-1})$, so it will move to state $(x_1 + 1, \vec{x}_{-1})$. More generally, for $k \in [1, d]$, suppose that we extend r by requests (root plus leaves) $1, 2, \ldots, k$. Then since $\varphi(x_1+1, \ldots, x_{k-1}+1, x_k-1, x_{k+1}, \ldots, x_d) > \varphi(\vec{x})$, the online algorithm will end up at state $(x_1+1, \ldots, x_{k-1}+1, x_k+1, x_{k+1}, \ldots, x_d)$. If we then extend r by the sequence $1, 2, \ldots, d$, the online algorithm will move to state $\vec{x} + \vec{1}$. The first state has root unfairness $-\sum_1^d x_i$ and the second state has root unfairness $d - \sum_1^d x_i$, so one of those two numbers is greater than or equal to $\lceil d/2\rceil$ in absolute value.

For the second part, we show that any G can be oriented so that the indegree and outdegree of every vertex differ by at most 1. We then run the online algorithm that serves requests for the edge $\{i, j\}$ by alternatingly having i as a driver and j as a driver, starting with i iff the edge is directed from j to i. ◀

▶ **Theorem 2.2.** *Consider the Greedy algorithm and assume that ties are broken by an adversary. Then for any connected request graph, there exists a request sequence for which Greedy has unfairness at least* $\lfloor n/2\rfloor$.

Proof. (sketch) The proof is by induction, restricting requests to a subgraph that is a tree. The idea is to increase the spread of the values by taking an edge $\{i, j\}$ between a subtree with low average unfairness and a subtree with high average unfairness. Then perform requests in the first subtree to maximize x_i, in the second subtree to minimize x_j. Then, if $x_i > x_j$, requesting edge $\{i, j\}$ will result in an even greater unbalance between the two subtrees. ◀

3 Random requests

3.1 Random requests on bounded genus graphs

We first prove the result for a star.

▶ **Theorem 3.1.** *Let G be a star with d leaves, root r and suppose that the requests are uniformly random. Then there is an algorithm such that for any time t, the unfairness at each leaf is at most one and the root has expected unfairness $O(1)$, with an exponential tail:* $\Pr(|Unfairness(r)| > k) \leq c\lambda^k$ *for some (c, λ) with $c > 0$ and $0 < \lambda < 1$.*

Sketch of proof of Theorem 3.1. The "star algorithm" is the following:

1. Every leaf $1 \leq i \leq d$ has a counter $x_i \in \{-1, 0, 1\}$. Initially, set $x_i = 0$ for all $i \in S_0 = \{1, \ldots, d/2\}$.[1] Set $x_i = 1$ for all $i \in S_1 = \{d/2 + 1, \ldots, 3d/4\}$, and set $x_i = -1$ for all $i \in S_{-1} = \{3d/4 + 1, \ldots, d\}$. The root r maintains a counter $x_0 = -\sum_1^n x_i$, which is initially equal to zero.

[1] For simplicity, we will assume that d is a multiple of four; this is not necessary.

2. When a random request (r, i) arrives, if $x_i \neq 0$ then the algorithm orients the edge so that $x_i = 0$. If $x_i = 0$ and $x_0 \neq 0$ then the algorithm orients the edge so that $|x_0|$ decreases. If $x_i = x_0 = 0$ then the choice is random.

To analyze this, observe that, in expectation, half of the leaves have value 0. If the root has unfairness x, then a request to an edge connecting the root to a non-zero leaf might increase $|x|$, but any request to an edge connecting the root to a 0 leaf reduces $|x|$. Hence, with some work, we obtain a proof that $|x|$ is bounded on average and that its distribution has an exponential tail. ◀

▶ **Corollary 3.2.** *Let G be a bounded genus graph on n vertices and suppose that the requests are random. Then there is a deterministic algorithm with average maximum unfairness $O(\log n)$.*

Proof. We partition the edges of G into stars so as to ensure that each vertex is a leaf of at most a constant number of stars, and the center of exactly one star. To handle a sequence of requests, for each star, handle the subsequence of the requests that are edges of that star using the algorithm from Theorem 3.1. ◀

3.2 Poor performance of *Global Greedy* for random requests on the line

Ajtai et al. [1] prove that a uniformly random sequence of requests in the complete graph induces a unfairness of $O(\log \log n)$ for *Global Greedy* with any tie breaking rule. We show that this is not necessarily true when the possible requests are restricted to edges in the social network.

▶ **Theorem 3.3.** *Consider a sequence of independent requests drawn uniformly at random when the graph is a line. Then the* Global Greedy *algorithm, with ties broken at random, has expected unfairness $\Omega\left((\log n/\log\log n)^{1/3}\right)$.*

To prove this theorem, we will need the following lemma.

▶ **Lemma 3.4.** *When the graph G is a line with n vertices, there exists a sequence of length $f(n) = \Theta(n^3)$ that creates maximum unfairness of $n/2$ for the* Global Greedy *algorithm with adversarial tie-breaking. After $f(n)/2$ steps of that sequence the maximum unfairness is already at least $n/8$.*

Proof. (of Theorem 3.3)

The idea is to consider multiple short segments of the line. The segments are of length k (to be determined below, about $\log^{1/3} n$), every segment has 2 extra vertices on the right, so there are about n/k different segments. We refer to the two extra vertices on the right of each segment as the "buffer zone" of the segment.

Consider a sequence of L random requests on the line, and focus on those, among those L requests, that fall into a particular segment and its buffer zone. A segment received Lk/n requests on average. Let L be such that $Lk/n = (3/4)f(k)$ and restrict attention to *good* segments, i.e. those segments that receive a number of requests in $[f(k)/2, f(k)]$.

With probability about $(1/k)^{k^3}$ the leftmost and rightmost buffer edges are never requested, and the requests to the segment follow the exact pattern of (a prefix of) the $f(k)$ requests required for the lower bound of Lemma 3.4. Conditioned upon this happening, the probability that the decisions made for tie breaking are as in Lemma 3.4 is at least $(1/2)^{f(k)}$, thus the probability that an unfairness of at least $k/8$ is reached in a particular good segment is $\geq (1/(2k))^{f(k)}$.

With probability at least $1/2$, the number of good segments is $\Omega(n/k)$, and by independence of the request sequence inside each segment, the probability that no segment gets unfairness $k/8$ is at most $\left(1-\frac{1}{(2k)^{f(k)}}\right)^{\Omega(n/k)}$. This is at most $1/e$ for $\Omega(n/k) > (2k)^{\Theta(k^3)}$. Taking the logarithm gives $\log n = \Theta(k^3)\log(k)$, hence $k = \Theta\left((\log n/\log\log n)^{1/3}\right)$. ◀

4 Lower bounds against Adaptive Adversaries

In this section we show an adaptive adversary which achieves an $\Omega(d)$ lower bound for any randomized algorithm, on a star with d leaves.

Let $V = \{i_1, i_2, \ldots, i_d\}$ be the leaves of the star and let r be the root. Define the subsets:

$$\begin{aligned} V^+ &= \left\{i_j \in V | x_{i_j} > 0\right\}, \\ V^- &= \left\{i_j \in V | x_{i_j} < 0\right\}, \\ V^0 &= \left\{i_j \in V | x_{i_j} = 0\right\}. \end{aligned}$$

▶ Remark. For simplicity, we assume that d is divisible by 4, but this is not necessary.

Our adversary generates a sequence, until either $|x_r| \geq d/4$ or there is a leaf i_j such that $|x_{i_j}| \geq d/4$. The sequence is generated as follows:

1. If there is a leaf i_j such that $v \in V^0$ then issue the request (r, i_j).
2. If $V^0 = \emptyset$ and $|V^+| \geq d/2$ then let $V^+ = \{i_1, \ldots, i_k\}$ such that $x_{i_j} \leq x_{i_{j+1}}$ and issue the requests (r, i_j) in order of increasing j. Stop after processing a request increases x_{i_j}.
3. If $V^0 = \emptyset$ and $|V^-| > d/2$ then let $V^- = \{i_1, \ldots, i_k\}$ such that $x_{i_j} \geq x_{i_{j+1}}$ and issue the requests (r, i_j) in order of increasing j. Stop after processing a request decreases x_{i_j}.

The following lemma is proved in [11].

▶ **Lemma 4.1.** *The request sequence generated by the adaptive adversary described above is well defined,* i.e.,

1. *Exactly one of the three cases happens at each iteration.*
2. *In case 2 either the unfairness of a leaf increases or $x_r > d/4$.*
3. *In case 3 either the unfairness of a leaf decreases or $x_r < -d/4$.*

For our analysis we define the potential function

$$\Phi(V) = \sum_{i \in V} d^{|x_i|}.$$

Note that this potential function does *not* take into account x_r.

▶ **Lemma 4.2.** *After each iteration of the adversary's decision loop the potential $\Phi(V)$ increases by at least $d-1$ or $|x_r| \geq d/4$.*

Proof. (sketch) We prove this by case analysis:

1. If there exists a leaf i_j such that $i_j \in V^0$ then the potential increases by exactly $d-1$.
2. If $|V^+| \geq d/2$ then from Lemma 4.1 the unfairness of one leaf was increased and at most the unfairness of $d-1$ leaves was decreased.
3. If $|V^-| > d/2$ then from Lemma 4.1 the unfairness of one leaf was decreased and at most the unfairness of $d-1$ leaves was increased.

In all three cases, the increase in potential caused by just one leaf is greater by more than $d-1$ than the decrease caused by the other leaves. ◀

▶ **Theorem 4.3.** *Assume that the social network is a star with d leaves. For any randomized algorithm, the adaptive adversary presented achieves unfairness $\Omega(d)$.*

Proof. From Lemma 4.1 the request sequence generated by the adversary is well defined and from Lemma 4.2, after each iteration of the adversary's decision loop either the potential $\Phi(V)$ increases by at least $d-1$ or $|x_r| \geq d/4$.

The initial potential is $\Phi(V) = d$. If after $(d \cdot d^{d/4-1})/(d-1) - 1$ iterations of the loop the inequality $|x_r| < d/4$ always holds then

$$\Phi(V) \geq d \cdot d^{d/4-1} - (d-1) + d \geq d \cdot d^{d/4-1} + 1.$$

So there must be at least one leaf with unfairness $\geq d/4$. ◀

5 Lower bounds against Oblivious Adversaries

5.1 Static Algorithms

Next, we consider *static algorithms* and bound the optimal unfairness that can be achieved by an algorithm in this class.

▶ **Definition 5.1.** A *state* is a vector $(x_i)_i$ where $x_i \in \mathbb{Z}$ represents the unfairness of vertex i. A randomized online algorithm is called *static* if there exists a probability distribution π over the set of states such that if the algorithm starts in π (*i.e.*, it starts at a state drawn according to π) then it remains in π after every possible request sequence.

Let $U(\vec{x})$ denote the maximum unfairness of state x. Then[2] the *expected maximum unfairness* of a static algorithm is

$$\sum_{\vec{x}} \pi(\vec{x}) \cdot U(\vec{x}).$$

As discussed in the introduction, *Randomized Local Greedy* preserves the distribution π in which each edge is oriented uniformly at random, and so *Randomized Local Greedy* is a static algorithm. The expected unfairness of every vertex i is $\Theta(\sqrt{d_i})$, where d_i is the degree of i, and the maximum unfairness is at most $O(\sqrt{n \log n})$ [1]. In particular, for the star with d leaves the unfairness of each leaf is at most 1, and the unfairness of the root is $\Theta(\sqrt{d})$.

▶ **Theorem 5.2.** *Assume G is the star with d leaves.*

1. *A slight variant of Local Greedy (Balanced Local Greedy) has optimal unfairness.*
2. *This value is $\Theta(\sqrt{d})$.*

Since states $\vec{x}$ that correspond to unfairness always satisfy $\sum_{i \in V} x_i = 0$, one of the coordinates can be inferred from the others. For compactness, we will drop the coordinate associated to the root and represent the state as a vector indexed by the leaves only.

To prove Theorem 5.2, we first state a property satisfied by the static distributions of static algorithms for the star with d leaves.

[2] There is a subtlety here in that the algorithm does not actually move initially to a state drawn from this distribution. Rather, it "pretends" to, and hence this definition of expected unfairness can be off by a factor of 2.

▶ **Lemma 5.3.** *Fix a static algorithm and let $\pi(\vec{x})$ be the corresponding static distribution. Fix a leaf j and $\vec{x}_{-j}$. Then the probability of states with even x_j must be equal to the probability of states with odd x_j:*

$$\sum_{k\in\mathbb{Z}} \pi(k,\vec{x}_{-i})(-1)^k = 0. \tag{2}$$

Proof. Suppose that the next request is the edge from root to leaf j. Conditioning on being on one of the states with given $\vec{x}_{-j}$, the state moves from even x_j to odd x_j and vice versa. Since the distribution before and after the request is the same, the two events must be equiprobable. ◀

[Proof of Theorem 5.2] We define an infinite linear programming relaxation P with one variable $\pi(\vec{x})$ for each $\vec{x} \in \mathbb{Z}^d$. By Lemma 5.3, the expected unfairness of any static algorithm is at least

$$\min \sum_x \pi(\vec{x})U(\vec{x}) \text{ s.t. } \quad \begin{cases} \sum_{\vec{x}} \pi(\vec{x}) & = 1 \\ \sum_k \pi(k,\vec{x}_{-i})(-1)^k & = 0 \quad \forall i \forall \vec{x}_{-i} \end{cases}$$

Consider a vector $\vec{x}$ such that $x_i = 0$. Then, by elimination, the second constraint determines $\pi(\vec{x})$ in terms of the variables $\pi(k,\vec{x}_{-i})$ for $k \neq 0$, and the vectors $(k,\vec{x}_{-i})$ have one fewer zero coordinate than $\vec{x}$. Extending this by induction on the number of non-zero coordinates of $\vec{x}$, we show in Claim 5.4 below that each variable $\pi(\vec{x})$ with at least one zero coordinate in $\vec{x}$ can be expressed as a linear combination of the variables $\pi(\vec{y})$ with $\vec{y} \in (\mathbb{Z}_{\neq 0})^d$.

▶ **Claim 5.4.** *Let $Q(\vec{x}) = \{\vec{y} : x_i(y_i - x_i) = 0$ and $y_i \neq 0$, for all $i\}$. Then for all $\vec{x}$*

$$\pi(\vec{x}) = (-1)^{\#0(\vec{x})} \sum_{\vec{y}\in Q(\vec{x})} \pi(\vec{y})(-1)^{\sum_i y_i - x_i}. \tag{3}$$

Moreover, the set of Equations (3) for all $\vec{x} \notin (\mathbb{Z}_{\neq 0})^d$ are equivalent to the set of Equations (2) in the linear program P.

Proof. We transform Equations (2) to the form of Equations (3), inductively by the number of zero entries of $\vec{x}$.

The base case, in which $\vec{x}$ has no 0's is vacuous. For the induction step, assume that Equation (3) holds for all $\vec{x}$ that have at most k 0's. Consider some $\vec{x}$ that has $k+1$ 0's: $\vec{x} = (0,\vec{x}_{-i})$ for some x_{-i} that has k 0's. Solving (2) for $\pi(\vec{x})$ we get

$$\begin{aligned} \pi(0,\vec{x}_{-i}) &= -\sum_{y_i\neq 0} \pi(y_i,\vec{x}_{-i})(-1)^{y_i} \\ &= -\sum_{y_i\neq 0} (-1)^{\#0(\vec{x}_{-i})} \sum_{\vec{y}_{-i}\in Q(\vec{x}_{-i})} \pi(y_i,\vec{y}_{-i})(-1)^{\sum_{k\neq i} y_k - x_k}(-1)^{y_i} \\ &= (-1)^{\#0(\vec{x})} \sum_{\vec{y}\in Q(\vec{x})} \pi(\vec{y})(-1)^{\sum_i y_i - x_i}. \end{aligned}$$

Notice that there are $k+1$ different equations in the set of Equations (2), one for each 0 in $\vec{x}$, that are transformed into the same equation. ◀

To simplify the linear program P, substitute the right hand side of every equality of the form given in Equation (3) for $\pi(\vec{x})$ (for all $\vec{x} \notin (\mathbb{Z}_{\neq 0})^d$) into the constraint $\sum_{\vec{x}} \pi(\vec{x}) = 1$. This yields

$$\sum_{\vec{y}\notin(\mathbb{Z}_{\neq 0})^d} \alpha_{\vec{y}}\pi(\vec{y}) = 1, \tag{4}$$

for some constants $\{\alpha_{\vec{y}}\}$.

By construction, any solution to this equation can be extended to a solution to the set of Equations (2). This allows us to reduce the linear programing relaxation to one with a single constraint. Therefore it has an optimal solution with only one non-zero variable. Let $\pi(\vec{y}^*)$ denote this variable. By substituting back to (3), we get that $\pi(\vec{x})$ is zero unless

$$\vec{x} \in H := \{\vec{z} \quad \text{s.t. for all } i \quad z_i \in \{0, y_i^*\}\}.$$

Moreover, since for $\vec{x} \in H$, there is only one non-zero term on the right hand side of (3), we can conclude that $\forall \vec{x} \in H$, either $\pi(\vec{x}) = \pi(\vec{y}^*)$ or $\pi(\vec{x}) = -\pi(\vec{y}^*)$.

Next, observe that all coordinates of $\vec{y}^*$ must be odd. Indeed, if some y_i^* is even, then (3) implies that for every vector $\vec{x} \in H$ we have $\pi(y_i^*, \vec{x}_{-i}) = -\pi(0, \vec{x}_{-i})$, which implies that the sum of π on the vectors in H is 0, a contradiction. Thus, all coordinates of $\vec{y}^*$ are odd, and so $\pi(\vec{x}) = \pi(\vec{y}^*)$ for all $\vec{x} \in H$. Since these probabilities sum to 1, they must all be equal to $1/2^d$.

Let $|\vec{x}| = |\sum_i x_i|$. The expected unfairness of the root is

$$\sum_{\vec{x}} \pi(\vec{x})|\vec{x}| = \sum_{x \in H} \pi(\vec{x})|\vec{x}| = \sum_{x \in \{0, y_i^*\}^d} \frac{1}{2^d}|\vec{x}| = \mathrm{E}[|\sum_i y_i^* X_i|],$$

where X_i are 0-1 unbiased Bernoulli random variables. The following claim shows that this quantity is minimized when exactly half ($\lfloor d/2 \rfloor$, to be precise) of the y_i^*'s are 1 and the rest are -1, exactly as in the case of *Balanced Local Greedy*.

▶ **Claim 5.5.** *Let $y_1^*, \ldots, y_d^*$ be odd integers and $X_1, \ldots, X_n$ be independent unbiased Bernoulli variables. The expectation*

$$E[|\sum_i y_i^* X_i|]$$

is minimized when half ($\lfloor d/2 \rfloor$, to be precise) of the y_i^*'s are -1 and the remaining are +1. This minimum value is $\Theta(\sqrt{d})$.

It follows from the theorem that the optimal unfairness among static algorithms is at least equal to the unfairness at the root of the *Balanced Local Greedy*. Since its unfairness on the leaves is at most one, *Balanced Local Greedy* has almost optimal unfairness among the static algorithms, within an additive term of 1. In fact, the additive term can be reduced to $O(1/\sqrt{d})$.

The main result of this section dashes any hopes to find a static algorithm with small unfairness. However, many natural algorithms—among them the *Global Greedy* algorithm—are not static and we hope that they will be shown to have small unfairness (substantially less than $O(\sqrt{d})$).

5.2 Bounds on *Randomized Global Greedy*

A very natural algorithm, *Randomized Global Greedy* is a candidate algorithm to give small competitive ratios against an oblivious algorithm. This algorithm is greedy, choosing the driver with the lower number of times to her credit, and breaking ties at random. The difference between *Randomized Global Greedy* and *Randomized Local Greedy* is that *Randomized Local Greedy* only uses randomization initially to determine the initial state and then alternates drivers whenever the same pair reappear.

Previously, the lower bound on the competitive ratio of *Randomized Global Greedy* was $\Omega(\log^{1/3} n)$ (on a clique) whereas the upper bound was $O(n)$. We improve the lower bound to $\Omega(\log n)$. To do this we make use of a potential function, the sum over drivers of the differences between the number of trips minus the number of times that the driver drove.

The key idea is to repeatedly generate unfairness and "push" it to some target set of drivers. This process works essentially as follows:

1. Apply a sequence of requests to generate potential in some set of drivers.
2. "Push" the potential to a target set of drivers.
3. Repeat.
4. Eventually, the target set will have high potential, which implies high unfairness.

We present here a sketch of the lower bound proof, full details of this process can be seen online at [11]. In this MSc thesis (of one of the authors) one can also see similar techniques applied to other settings.

First, we begin with a definition of the potential function.

▶ **Definition 5.6.** Let $i_1, i_2, \ldots, i_n$ be n vertices where x_{i_j} is the unfairness of vertex i_j. Assume that there is a value $u \in \mathbb{Z}$ such that for all $1 \leq j \leq n$ it holds that $x_{i_j} \in \{u-1, u, u+1\}$. Define the potential with respect to base unfairness u as

$$\Phi_u(\{x_{i_1}, x_{i_2}, \ldots, x_{i_n}\}) = \sum_{1 \leq j \leq n} (x_{i_j} - u).$$

Now that we have defined the potential function, we define "pushing".

▶ **Definition 5.7.** Let $v', v'', r \in V$ be distinct vertices and let $S = \{i_1, i_2, i_3, \ldots, i_n\}$ be a set of n vertices such that $v', v'', r \notin S$.

Let $\alpha \in \mathbb{N}$ be a large number, as a function of n. The sequence $\text{push}_u(\{v', v''\}, S, r)$ is composed of three subsequences:

$$\begin{aligned}
&\text{push}_u^{s1}(\{v', v''\}, S, r) = (r, v'), (r, v''),\\
&\text{push}_u^{d}(\{v', v''\}, S, r) = (r, i_n), (r, i_n), (r, i_{n-1}), (r, i_{n-1}), \ldots, (r, i_1), (r, i_1),\\
&\text{push}_u^{s2}(\{v', v''\}, S, r) = (r, v''), (r, v').
\end{aligned}$$

And,

$$\begin{aligned}
\text{push}_u(\{v', v''\}, S, r) = \big(&\text{push}_u^{s1}(\{v', v''\}, S, r) \| \\
&\text{push}_u^{d}(\{v', v''\}, S, r) \| \text{push}_u^{s2}(\{v', v''\}, S, r)\big)^{\alpha}.
\end{aligned}$$

▶ Remark. Notation: (v, w) is a shorthand for requesting that v and w carpool together.

We use the following property thats holds after the *Randomized Global Greedy* algorithm processes $\text{push}_u(\{v', v''\}, S, r)$:

Define $\Phi_{\text{init}}(S) = \Phi(S)$ and $\Phi_{\text{init}}(\{v', v''\}) = \Phi(\{v', v''\})$ as the potentials before processing the sequence $\text{push}_u(\{v', v''\}, S, r)$. Define $\Phi_{\text{end}}(S) = \Phi(S)$ and $\Phi_{\text{end}}(\{v', v''\}) = \Phi(\{v', v''\})$ as the potentials after processing the sequence.

▶ **Lemma 5.8.** *Assume that: (a) For some $u \in \mathbb{Z}$, $x_r = u$, (b) That $x'_v, x''_v \in \{u-1, u+1\}$, and (c) For all $i \in S$, $x_i \in \{u-1, u+1\}$.*

If $|(x_{v'} - u) + (x_{v''} - u) + \Phi_{\text{init}}(S)| \leq |S|$ then, with high probability, after the Randomized Global Greedy *algorithm processes the sequence $\text{push}_u(\{v', v''\}, S, r)$ the equality $\Phi_{\text{end}}(S) = \Phi_{\text{init}}(S) + \Phi_{\text{init}}(\{v', v''\})$ holds.*

Proof. (sketch)

Observe that if $x_{v'} = u+1$, $x_{v''} = u+1$ and $x_r = u$ then after processing the subsequence $(r, v'), (r, v'')$, with probability $1/2$, $x_r = u+2$. Now observe that if $x_r = u+2$ and for some j, $x_{i_j} = u-1$ then after processing the two requests $(r, i_j), (r, i_j)$ it holds that $x_{i_j} = u+1$ and $x_r = u$. Thus the potential of $\{v', v''\}$ was "pushed" into i_j.

A similar thing happens if $x_{v'} = u-1$, $x_{v''} = u-1$ and $x_{i_j} = u+1$. The constant α is chosen to be large such this occurs with high probability.

The full proof is available online in [11]. ◀

Now, using this push sequence we "accumulate" a large potential (in respect to base unfairness u) in a specific set A such that $|A| = O(n)$. *I.e.*, we define a sequence such that after *Randomized Global Greedy* processes it $Pr(|\Phi_u(A)| > |A/2|) > 1/2$. In essence, this is done by repeatedly using a sequence that creates unfairness in a small set G and "pushing" this unfairness into A. This is called "accumulation".

Another useful subsequence is that of "distillation", which takes the potential of a set and, with high probability, pushes it into a subset. Let r be a vertex and $S = \{i_1, i_2, i_3, \ldots, i_m\}$ be a set of m vertices such that m is even and $r \notin S$. The $\text{distill}_u(S, r)$ event sequence is

$$\begin{aligned}\text{distill}_u(S, r) = \text{push}_u(\{i_1, i_2\}, \{i_3, \ldots, i_m\}, r) \| \text{push}_u(\{i_3, i_4\}, \{i_5, \ldots, i_m\}, r) \| \\ \cdots \| \text{push}_u(\{i_{m-3}, i_{m-2}\}, \{i_{m-1}, i_m\}, r).\end{aligned}$$

Define the tail of S, S_ℓ, as $S_\ell = \{i_k \in S | k \geq \ell\}$.

▶ **Lemma 5.9.** *With high probability, after the* Randomized Global Greedy *algorithm processes the sequence* $\text{distill}_u(S)$

$$\Phi_u(S_{|S|-|\Phi_u(S)|+1}) = \Phi_u(S).$$

Proof. (sketch) This stems from repeatedly applying Lemma 5.8 to increasingly smaller sets. The full proof is available online in [11]. ◀

Now using the sequences "accumulation" and "distillation" we are able to create (with probability greater than $1/2$) a set A such that $|A| = O(n)$ and $|\Phi_u(A) = |A|$. The latter is equivalent to all the unfairnesses of vertices in A being equal to either $u-1$ or $u+1$.

Now split A into three different sets, A_1, A_2, and A_3 and do "accumulation" and "distillation" to each set individually. Now, with some probability, one of these subsets has unfairnesses which are all equal to either $u-2$ or $u+2$. Repeat this process. In [11] we prove that this can be repeated $\Omega(\log n)$ times with constant probability. And thus the following theorem is proved:

▶ **Theorem 5.10.** *The sequence above achieves an expected unfairness of* $\Omega(\log n)$ *for the* Randomized Global Greedy *algorithm when run on a clique.*

6 Open Questions

The outstanding open questions that follow immediately from this work are:

- Is there any randomized algorithm with unfairness $o(\sqrt{d})$ on the star with d leaves?
- Does *Randomized Global Greedy* have $o(n)$ unfairness on the star or on the line?

At this point we have no non-trivial upper bound on the star. The best algorithm we know is *Randomized Local Greedy*, which achieves $\sqrt{n}$ unfairness.

Acknowledgments

We are grateful to Nimrod Fiat and Clemens von Stengel for their help during the early stages of this work.

The second author acknowledges the support of NSF grant CCF 1420381. The third author acknowledges the support of ERC Advanced Grant 321171 (ALGAME).

References

1 Miklos Ajtai, James Aspnes, Moni Naor, Yuval Rabani, Leonard J Schulman, and Orli Waarts. Fairness in scheduling. *Journal of Algorithms*, 29(2):306–357, 1998.

2 Amihood Amir, Oren Kapah, Tsvi Kopelowitz, Moni Naor, and Ely Porat. The family holiday gathering problem or fair and periodic scheduling of independent sets. *CoRR*, abs/1408.2279, 2014. URL: http://arxiv.org/abs/1408.2279.

3 Joao Pedro Boavida, Vikram Kamat, Darshana Nakum, Ryan Nong, Chai Wah Wu, and Xinyi Zhang. Algorithms for the carpool problem. *IMA Preprint Series*, pages 2133–6, 2006.

4 Don Coppersmith, Tomasz Nowicki, Giuseppe Paleologo, Charles Tresser, and Chai Wah Wu. The optimality of the online greedy algorithm in carpool and chairman assignment problems. *ACM Trans. Algorithms*, 7(3):37:1–37:22, July 2011. URL: http://doi.acm.org/10.1145/1978782.1978792, doi:10.1145/1978782.1978792.

5 Ronald Fagin and John H Williams. A fair carpool scheduling algorithm. *IBM Journal of Research and development*, 27(2):133–139, 1983.

6 Ernst Fehr and Klaus M Schmidt. A theory of fairness, competition, and cooperation. *Quarterly journal of Economics*, pages 817–868, 1999.

7 L. Festinger. *A Theory of Cognitive Dissonance*. Mass communication series. Stanford University Press, 1962. URL: https://books.google.co.il/books?id=voeQ-8CASacC.

8 George F. Loewenstein, Leigh L. Thompson, and Max H. Bazerman. Social utility and decision making in interpersonal contexts. *Journal of Personality and Social Psychology*, 57(3):426–441, 1989. URL: http://dx.doi.org/10.1037/0022-3514.57.3.426, doi:10.1037/0022-3514.57.3.426.

9 Moni Naor. On fairness in the carpool problem. *Journal of Algorithms*, 55(1):93 – 98, 2005. URL: http://www.sciencedirect.com/science/article/pii/S0196677404000781, doi:http://dx.doi.org/10.1016/j.jalgor.2004.05.001.

10 R. Tijdeman. The chairman assignment problem. *Discrete Mathematics*, 32(3):323 – 330, 1980. URL: http://www.sciencedirect.com/science/article/pii/0012365X80902691, doi:http://dx.doi.org/10.1016/0012-365X(80)90269-1.

11 Zach and Fiat. Lower bounds for carpooling. https://www.cs.tau.ac.il/~fiat/rotem_msc_thesis.pdf, 2015.