TEL AVIV UNIVERSITY אוניברסיטת תל-אביב
The Raymond and Beverly Sackler Faculty of Exact Sciences
The Blavatnik School of Computer Science

Lower Bounds for Carpooling

Thesis submitted in partial fulfillment
of graduate requirements for the degree
"Master of Sciences" in Tel Aviv University
School of Computer Science

By
Rotem Zach

Prepared under the supervision of Prof. Amos Fiat

August 2015

Abstract

Imagine a group of people where each day a pair of them want to go to the same place (e.g., their place of work). Each person in the group owns a car, but they prefer to save money and go together. How will they decide who will drive, and who will be the passenger? This is called the "carpool" problem.

We look at the online case, where it is not known in advance who will be in each pair. The algorithm must make decisions with only knowledge of previous events and of the current event. In this paper, we concentrate on the case where the event sequence is generated by an oblivious adversary, one that knows the algorithm, but does not have knowledge of the algorithm's random coin tosses (if there are any).

How is the performance of an algorithm measured? After all the events have been processed, define the unfairness of each person as the absolute value of the difference between the number of times the person has driven and the amount of times they have been a passenger. The algorithm must minimize this value for each person in the group.

There are two different scenarios that are of interest, one where every pair is possible and one where there is a fixed rider every day and only the identity of the other rider changes.

We show three lower bounds:

- For the case with a fixed rider, we prove an $\Omega(n)$ lower bound for any algorithm, in the case that the event sequence is generated by an adaptive adversary.

- For the case with a fixed rider, there exists an algorithm which achieves a constant expected unfairness when the events are uniform. We show an $\Omega(\sqrt{n})$

lower bound in the case where the events are determined by an oblivious adversary.

- For the case where every pair is possible (*i.e.*, there is no fixed rider), there exists a natural randomized greedy algorithm. We show an $\Omega(\log n)$ lower bound for this algorithm if the events are generated by an oblivious adversary.

All the lower bounds shown are constructive, *i.e.*, we show an event sequence that achieves these bounds.

Table of Contents

*

Acknowledgments

First, I would like to thank my supervisor Prof. Amos Fiat. Amos has given me a wonderful opportunity to discover the world of research.

He has been very supportive of me and has helped with anything needed. We have met many Fridays and Saturdays to talk about this research. Thank you very much!

Next, I would like to thank Claire Mathieu, Elias Koutsoupias and Anna Karlin with whom I have had many fruitful exchanges regarding the carpooling problem in general and specifically the lower bounds shown in this paper.

I Introduction

In multiple experimental studies involving hundreds of graduate students, Loewenstein, Thompson, and Bazerman [7] 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 [4] coined the phrase *inequity aversion* to describe this phenomena. Festinger [5] had much earlier introduced the concept of cognitive dissonance, and inequity aversion is modelled as a special case thereof. Supposedly, inequity aversion may lead individuals to make significant changes, including stopping interpersonal relationships where inequities arise.

The interest in online real-time services that allow drivers and passengers to organize carpooling has been increasing in the last several years. Just looking at the amount of carpooling applications available for smartphones suggests that there is interest in this field.

One of the applications available is "Carma" [2] (formerly known as "Avego"). They provide a service that pairs drivers and passengers in selected metropolitan areas. The driver is allowed to collect a small fee for each mile driven.

Even one of the biggest technology companies has recently shown interest. In July 2015, the Google owned Waze has announced a ride-sharing service [10]. This application pairs drivers and passengers that are taking a similar route and is currently (August 2015) available in Israel's Tel Aviv area.

The main theoretical model of interest was introduced by Fagin and Williams [3]. This is a stylized mathematical model in which one can study questions related to minimizing inequity. As described in [3], *"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. [1]

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 $F_i(t) = \sum_{\tau \leq t} f_i(\tau)$, 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

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

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

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

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 (see, e.g. [9].)

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

[1] We remark that this notion of equity is not that from interactions between Tom and Jerry both are (approximately) equally well off. The notion here is global, taking all their interactions into account. In total, Tom and Jerry should be approximately equal in payoff.

additions[2], 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[3]

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

Ajtai et al. obtained results for two different online settings: when the events (carpools) are selected at random, and when the events 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 event (i, j), the driver among i and j with minimum unfairness drives; in case of a tie, the choice is arbitrary. For a uniformly random event 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 event sequence. They were able to obtain a better upper bound[4] 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 $\Theta(\sqrt{n \log n})$. Finally, they proved that every randomized algorithm has maximum unfairness equal to $\Omega\big((\log n)^{1/3}\big)$.

[2] We will call these edge additions *events*.

[3] Note that $\text{indegree}(i) - \text{outdegree}(i) = 2(D_i(t) - F_i(t))$. Dropping the factor of $1/2$ in defining the unfairness of a driver simplifies the discussion slightly.

[4] *Randomized Global Greedy*, the version of *Global Greedy* in which ties are broken at random, is conjectured to be much better, perhaps even $\text{polylog}(n)$.

1.1 Our Results

In paper [6] we (Fiat, Karlin, Koutsoupias, Mathieu and Zach) take a different approach, studying the carpool problem in the context of social networks. The social network context for the carpool problem is the setting where the people involved belong to a social network G, and every event (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 [3, 1] can be seen as studying the special case where the social network is a clique.

Let G be a social network with n vertices, and of maximum degree d. We show that it is enough to look at the case where G is a star with d leaves. We prove that for every deterministic algorithm there exists a sequence resulting in unfairness $\lfloor d/2 \rfloor$ and showed that this is tight. In the case the events are random, *Global Greedy* has expected unfairness at least $\Omega((\log n/\log\log n)^{1/3})$. Also an analysis was given for *static algorithms*. Static algorithms form a very natural class of randomized online algorithms. Intuitively, they render an adversary powerless to construct a bad event sequence: every event sequence performs the same against such an algorithm. It is shown that every randomized static algorithm has unfairness $\Omega(\sqrt{d})$ and therefore, *Randomized Local Greedy* is essentially optimal among static algorithms.

In the full paper [6] the *Star Algorithm* is introduced and a $O(1)$ bound on the unfairness is shown where in the case that the graph is a star and the events are randomly uniform. From this, bounds of $\Omega(\log d)$ for a graph of bounded degree d and $\Omega(\log n)$ for a graph with bounded genus (e.g. a planar graph) are derived.

In this thesis I will not cover all of [6] but only the lower bounds in which I was most involved.

1.1.1 Adaptive Adversaries on the Star

We present an adaptive adversary that achieves unfairness of at least $n/4$ for *any randomized algorithm*, where n is the number of leaves.

1.1.2 Star Algorithm *with Oblivious Adversaries on the Star*

We show a sequence that achieves a lower bound of $\Omega(\sqrt{n})$. This hints that, in essence, this algorithm is no better than any static algorithm when the events are generated by an oblivious adversary.

1.1.3 Randomized Global Greedy *with Oblivious Adversaries on the Clique*

Ajtai et al [1] first presented the *Randomized Global Greedy* algorithm, and proved an $O(\log n)$ upper bound when the events are uniformly random. We show a sequence that achieves a lower bound of $\Omega(\log n)$, and thus prove that *Randomized Global Greedy* is $\Omega(\log n)$ with an oblivious adversary. This is a new result, previously no lower bound was known for *Randomized Global Greedy* better than the $\Omega(\log^{1/3} n)$ bound known for any randomized algorithm.

1.2 Notation

In what follows, we often suppress the dependence on t in our notation for the unfairness of driver i at time t. We label our vertices with natural numbers, specifically, we use i to denote the driver i and $u(i)$ to denote the unfairness of driver i at time t (*i.e.*, $u(i) := 2D_i(t) - F_i(t) = \text{indegree}(i) - \text{outdegree}(i)$), where t is understood. (Note that since $u(i)$ is equal to $\text{indegree}(i) - \text{outdegree}(i)$, it holds that $\sum_i u(i) = 0$ at all times.)

II Problem Statement

2.1 *The* Carpool *Problem*

First we define the *carpool* problem formally. Let there be n vertices, $V = \{1, 2, \ldots, n\}$ where each vertex represents a "rider". Each day t there is a event $\sigma_t = (v_t, w_t)$ where $v_t, w_t \in V$ and $v_t \neq w_t$. The *algorithm* must decide which of v_t and w_t will drive. Denote d_t as the driver on day t, and denote p_t as the passenger on day t.

Let $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_T$ be a sequence of events spanning T days. The unfairness $u_\sigma(v)$ of vertex v over the event sequence σ is defined as

$$u_\sigma(v) = |\{1 \leq t \leq T | d_t = v\}| - |\{1 \leq t \leq T | p_t = v\}|$$

Throughout this paper σ is implied so we use the notation $u(v)$ instead of $u_\sigma(v)$.

The goal of the algorithm is to minimize

$$\max_\sigma \max_v |u_\sigma(v)|.$$

This value is called the unfairness of the algorithm.

2.2 *Social Networks*

We consider a new model for the *carpool* problem which restricts the set of possible events to a graph G. Thus only events of the form $\sigma_t = (v_t, w_t)$ where the edge $(v_t, w_t) \in E(G)$ are possible.

A very simple example of a social network is the "star", *i.e.*, there is a distinguished root r which must ride every day. E.g., we can state that for every t, $v_t = r$.

An adversarial lower bound on the star can easily be extended to a bound on a graph with maximum degree d. Let v_d be a maximal degree vertex. The vertex v_d and its adjacent edges induce a subgraph which is a star with d leaves. Run the adversary on this graph. Thus a bound of $\Omega(f(n))$ on the star becomes a bound of $\Omega(f(d))$ on a graph with maximum degree d.

In the context of social networks, the general case where every event (v_t, w_t) such that $v_t, w_t \in V$ is possible can be viewed as a clique, so $G = K_n$.

2.3 Offline vs. Online

Fagin and Williams [3] present the offline version of the problem, *i.e.*, the algorithm knows the whole sequence of events σ beforehand and when deciding who will be the driver the algorithm has full knowledge of all past and future events. There is an algorithm that guarantees maximum unfairness of 1 in this setting.

The online version of the problem is when only the previous and current event are known at time t and the algorithm must make a decision who will drive before being revealed the event σ_{t+1}. In this paper we solely consider the online case.

2.4 Input Settings

Three main types of input settings have been studied:

1. Stochastic Uniform Model - each event σ_t is uniformly and independently distributed over all possible events $(v_t, w_t) \in G$.

2. Oblivious Adversary - the adversary who generates the sequence has full knowledge of the algorithm but not of the instantiation of the random coin tosses made.

3. Adaptive Adversary - the adversary has full knowledge of the algorithm and learns the random coin tosses made by the algorithm, but only after assign-

ments are made. An adaptive adversary considers all previous events issued and all previous actions made by the algorithm.

In this thesis we focus on the case where the events are generated by oblivious and adaptive adversaries.

2.5 Fairness

In this subsection, we leave the 2-player scenario and look at the generalized carpool problem in which at most $k > 0$ (and not necessarily 2) players want to carpool each day. The fairness notion described above was first presented by Fagin and Williams [3].

For a set V of n players define the event $\sigma_t = (v_1, v_2, \ldots, v_{n(t)})$ where $0 < n(t) \leq k$ is the number of players that want to carpool on day t. The event sequence spanning T days is $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_T$.

Using this terminology the number of times player v should drive is optimally

$$\sum_{\{1 \leq t \leq T | v \in \sigma_t\}} \frac{1}{n(t)}.$$

If for all t, $n(t) = 2$, each player should drive half the days they participate. This is the fairness notion used in this paper.

Naor [8] takes an axiomatic approach to defining fairness. Let $\phi_v(\sigma)$ be the "fair share" of player v for event sequence σ. An algorithm should minimize the absolute difference between the number of times player v drives and $\phi_v(\sigma)$. The four requirements of ϕ are:

1. Full Coverage: The sum of the fair shares of all players equals the total number of days in the event sequence, *i.e.*, $\sum_{v \in V} \phi_v(\sigma) = T$.

2. Symmetry: If two players have exactly the same schedule, that is they appear on the same set of days, then their fair shares should be the same.

3. Dummy: A player that never shows on any day has fair share 0.

4. Concatenation: Given two event sequences σ_1, σ_2 consider the event sequence $\sigma = \sigma_1 \| \sigma_2$. The fair share of each player v for σ is the sum of the fair shares of v for σ_1 and σ_2, *i.e.*, $\phi_v(\sigma) = \phi_v(\sigma_1) + \phi_v(\sigma_2)$.

Naor proves that the only function satisfying these requirements is the one given by Fagin and Williams.

2.6 Expected Unfairness

We need to define unfairness for randomized algorithms. Let σ be an event sequence, π the random coin tosses generated by the algorithm. The *expected unfairness* over σ is

$$E_\pi \Big[\max_v |u_\sigma(v)| \Big].$$

And thus the goal of the algorithm is to minimize

$$\max_\sigma E_\pi \Big[\max_v |u_\sigma(v)| \Big].$$

2.7 Bounds

Tables 2.1 and 2.2 summarize old and new results under various input assumptions and various underlying graphs. Inputs are either stochastic uniform sequences, generated by oblivious adversaries, or generated by adaptive adversaries. The underlying graphs are the clique, the star network, and graphs of bounded degree/genus.

We (Fiat et al.) present these new bounds in paper [6]. In this thesis, I only present the lower bounds in which I was most involved.

First we show a $\Theta(n)$ lower bound for any randomized algorithm on the star if the event sequence is generated by an adaptive adversary.

Setting	RGG on the Clique	RLG on the Clique	Det. Alg. on the Clique
Stochastic Uniform	$\Theta(\log \log n)$ [1]	$O(\sqrt{n \log n})$ [1]	$\Theta(\log \log n)$ [1]
Oblivious Adversary	$\Omega(\log^{1/3}(n))$ [1] $\Omega(\log n)$ Theorem 6.2.14	$O(\sqrt{n \log n})$ [1]	$\Theta(n)$ [1]
Adaptive Adversary	$\Theta(n)$ [1]	$\Theta(n)$ [1]	$\Theta(n)$ [1]

Table 2.1: Previous and new bounds on the competitive ratios obtained by *Randomized Global Greedy* (RGG), *Randomized Local Greedy* (RLG) and deterministic algorithms in various settings.

Setting	*Star Algorithm* on the Star	RGG on the Star	Random Alg. on the Star
Stochastic Uniform	$\Theta(1)$ [6]	$\Omega((\frac{\log n}{\log \log n})^{1/3})$ [6]	???
Oblivious Adversary	$\Omega(\sqrt{n})$ Theorem 5.2.9	???	???
Adaptive Adversary	$\Theta(n)$ Theorem 3.2.5	$\Theta(n)$ Theorem 3.2.5	$\Theta(n)$ Theorem 3.2.5

Table 2.2: The bounds shown in this paper for the different algorithms on the star. RGG is *Randomized Global Greedy*.

Setting	RGG on the line	Det. Alg., bounded genus graphs	Det. Alg., max. degree d
Stochastic Uniform	$\Omega((\frac{\log n}{\log \log n})^{1/3})$ [6]	$O(\log n)$ [6]	???
Oblivious Adversary	$\Omega((\frac{\log n}{\log \log n})^{1/3})$ [6]	???	$\Theta(d)$ [6]

Table 2.3: The bounds shown in this paper for different social networks. RGG is *Randomized Global Greedy*.

Also, we show that while *Randomized Global Greedy* has a large asymptotic bound on its expected unfairness when run on the star for uniform adversaries, there exists an algorithm, the *Star Algorithm*, which achieves a constant expected unfairness. We also show that, unfortunately, this algorithm has a large expected unfairness for oblivious adversaries, $\Omega(\sqrt{n})$.

Finally, we improve the known $\Omega(\log^{1/3} n)$ bound for *Randomized Global Greedy* for oblivious adversaries when run on the clique, and show that it is $\Omega(\log n)$.

In this thesis we only present the new bounds for oblivious and adaptive adversaries, and not those for stochastic uniform event sequences.

2.8 Oblivious Adversary Toolbox

A major contribution of this thesis is the techniques, or "toolbox", used to prove the oblivious adversary lower bounds. In Section 4 we present idealized versions of sequences, that after being processed manipulate the unfairness of the vertices in specific ways. In Sections 5 and 6 we give concrete examples of sequences which have these properties for, respectively, the *Star Algorithm* on the star and *Randomized Global Greedy* on the clique.

III Lower Bound of $\Omega(n)$ for Adaptive Adversaries on the Star

In this section we show a constructive $\Omega(n)$ lower bound for any randomized algorithm, when the social network is a star, in the adaptive adversary input setting.

3.1 Adversary

Let $V = \{1, 2, \ldots, n\}$ be the leaves of the star and let r be the root. We define an adaptive adversary and prove that it achieves a lower bound of $\Omega(n)$ for any randomized algorithm.

Recall that the unfairness $u(v)$ for $v \in S$ can be either *positive*,*negative* or *zero*.

Define the subsets:

$$\begin{aligned} V^+ &= \{v \in V | u(v) > 0\}, \\ V^- &= \{v \in V | u(v) < 0\}, \\ V^0 &= \{v \in V | u(v) = 0\}. \end{aligned}$$

Remark 3.1.1. For simplicity, we assume that n is divisible by 4, but this is not necessary.

Our adversary generates a sequence, until either $|u(r)| \geq n/4$ or there is a leaf v such that $|u(v)| \geq n/4$. The sequence is generated as follows:

1. If there is a leaf v such that $v \in V^0$ then issue the event (r, v).

2. If $V^0 = \emptyset$ and $|V^+| \geq n/2$ then let $V^+ = \{v_1, \ldots, v_k\}$ such that $u(v_j) \leq u(v_{j+1})$ and issue the events (r, v_j) in order of increasing j. Stop after processing an event increases $u(v_j)$.

3. If $V^0 = \emptyset$ and $|V^-| > n/2$ then let $V^- = \{v_1, \ldots, v_k\}$ such that $u(v_j) \geq u(v_{j+1})$ and issue the events (r, v_j) in order of increasing j. Stop after processing an event decreases $u(v_j)$.

3.2 Analysis

Lemma 3.2.1. *The event 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 $u(r) > n/4$.*

3. *In case 3 either the unfairness of a leaf decreases or $u(r) < -n/4$.*

Proof. Each leaf is in exactly one of V^+, V^-, V^0 thus $|V^+| + |V^-| + |V^0| = n$. So either $V^0 \neq \emptyset$ and the first case occurs. Or $|V^+| + |V^-| = n$ and either $|V^+| \geq n/2$ and the second case is encountered or $|V^-| > n/2$ and the third case happens.

We defined that the adversary stops if $|u(r)| \geq n/4$. Thus, before entering this iteration it holds that $|u(r)| < n/4$. If case 2 was entered, then $|V^+| \geq n/2$. If the unfairness of all the leaves decreased then subsequent to each (r, v_j) event the unfairness of the root increased by 1. We assumed that $u(r) > -n/4$ and thus the unfairness is now

$$u(r) > -\frac{n}{4} + \frac{n}{2} = \frac{n}{4}.$$

If case 3 occurred, then $|V^-| > n/2$. If the unfairness of all the leaves increased then subsequent to each (r, v_j) event the unfairness of the root decreased by 1. We assumed that $u(r) < n/4$ and thus the unfairness is now

$$u(r) < \frac{n}{4} - \frac{n}{2} = -\frac{n}{4}.$$

□

Define the potential function

$$\Phi(V) = \sum_{v \in V} n^{|u(v)|}.$$

Note that this potential function does *not* take into account the unfairness of the root.

Lemma 3.2.2. *After each iteration of the adversary's decision loop the potential* $\Phi(V)$ *increases by at least* $n-1$ *or* $|u(r)| \geq n/4$.

Proof. If $|u(r)| \geq n/4$ we are done. Assume that $|u(r)| < n/4$. Let $\Phi = \Phi(V)$ be the potential, $u(v)$ be the unfairness of leaf v before processing the events generated during the iteration. Let $\Phi' = \Phi(V)$ be the potential, $u'(v)$ be the unfairness of leaf v subsequent to processing the events generated during the iteration.

We prove this by case analysis:

1. If there exists a leaf v such that $v \in V^0$ then $|u'(v) - u(v)| = 1$ so $\Phi' - \Phi = n^1 - n^0 = n - 1$.

2. If $|V^+| \geq n/2$ then from Lemma 3.2.1 the unfairness of one leaf was increased and at most the unfairness of $n-1$ leaves was decreased.

3. If $|V^-| > n/2$ then from Lemma 3.2.1 the unfairness of one leaf was decreased and at most the unfairness of $n-1$ leaves was increased.

In cases 2 and 3 above, let the leaf whose value was changed be v_ℓ (remember the

order $\{v_1, \ldots, v_k\}$ such that $|u(v_j)| \leq |u(v_{j+1})|$). Thus

$$\begin{aligned}
\Phi' - \Phi &= \sum_{1 \leq j \leq \ell} \left[n^{|u'(v_j)|} - n^{|u(v_j)|} \right] \\
&= n^{|u(v_\ell)|+1} - n^{|u(v_\ell)|} + \sum_{1 \leq j \leq \ell-1} \left[n^{|u(v_j)|-1} - n^{|u(v_j)|} \right] \\
&\geq n^{|u(v_\ell)|+1} - n^{|u(v_\ell)|} + \sum_{1 \leq j \leq \ell-1} \left[n^{|u(v_\ell)|-1} - n^{|u(v_\ell)|} \right] \\
&\geq n^{|u(v_\ell)|+1} - n^{|u(v_\ell)|} + (n-1)\left[n^{|u(v_\ell)|-1} - n^{|u(v_\ell)|} \right] \\
&= n^{|u(v_\ell)|+1} - n \cdot n^{|u(v_\ell)|} + (n-1) n^{|u(v_\ell)|-1} \\
&= (n-1) n^{|u(v_\ell)|-1} \geq n - 1.
\end{aligned}$$

□

Theorem 3.2.3. *Assume that the social network is a star. For any randomized algorithm, the adversary presented achieves unfairness $\Omega(n)$.*

Proof. From Lemma 3.2.1 the event sequence generated by the adversary is well defined and from Lemma 3.2.2, after each iteration of the adversary's decision loop either the potential $\Phi(V)$ increases by at least $n-1$ or $|u(r)| \geq n/4$.

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

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

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

Remark 3.2.4. In [6] we present a non-constructive proof that the unfairness of any deterministic algorithm is at least $\lfloor n/2 \rfloor$ and we also present an algorithm that achieves this upper bound.

Theorem 3.2.5. *The carpool problem on the star is $\Theta(n)$ if the event sequence is generated by an adaptive adversary.*

Proof. In [6] we present a deterministic algorithm with an upper bound of $O(n)$ in the adversarial input setting.

From 3.2.3 any algorithm must have an $\Omega(n)$ bound. □

IV Oblivious Adversary Toolbox

Previous tools used to give lower bounds for online algorithms in the context of oblivious adversaries seem to consist of "guessing" a short sequence of random decisions, and effectively transforming an oblivious adversary into a deterministic adversary on a much smaller set of agents.

To obtain our lower bounds, we make use of some new techniques that may possibly be of independent interest. We define the notion of "set unfairness" and then present specific event sequences that manipulate this value. These can be seen as a generic "toolbox" to be used by oblivious adversaries.

4.1 Preliminaries

We use the following notation:

1. Let $V = \{1, 2, \ldots, n\}$ be the set of vertices.

2. Let $u(v)$ for $v \in V$ be the unfairness of the vertex v.

3. The unfairness vector of $S \subseteq V$, $S = \{v_1, v_2, \ldots, v_m\}$ is $\langle u(v_1), u(v_2), \ldots, u(v_n)\rangle$.

We extend the definition of unfairness of vertices to a notion of unfairness of sets. We call this the "potential" of a set.

Definition 4.1.1 (Set Potential)**.** Define the potential of a set $S \subseteq V$ with respect to a "base unfairness" $u \in \mathbb{Z}$ as

$$\Phi_u(S) = \sum_{v \in S} (u(v) - u).$$

Observation 4.1.2. *Let $u \in \mathbb{Z}$ be some value, let $S \subseteq V$ be a set of vertices, and let $\sigma = (v_1, w_1), (v_2, w_2), \ldots, (v_T, w_T)$ be a sequence of events such that for all $1 \leq t \leq T$, $v_t, w_t \in S$. Denote $\Phi_{\text{init}}(S) = \Phi_u(S)$ before the sequence σ is processed, $\Phi_{\text{end}}(S) = \Phi_u(S)$ after the sequence is processed. Then $\Phi_{\text{init}}(S) = \Phi_{\text{end}}(S)$.*

Another important notion is the "potential capacity" of a set.

Definition 4.1.3 (Potential Capacity)**.** Let $u \in \mathbb{Z}$ be a "base unfairness", $\mathcal{A}$ be an algorithm and $S \subseteq V$ be a set of vertices. Define $\text{cap}_u^{\mathcal{A}}(S)$, the "potential capacity" of S, to be the maximum absolute value of $\Phi_u(S)$. *I.e.*, for any sequence of events σ, the absolute value of the potential, $\Phi_u(S)$, after $\mathcal{A}$ processes σ, is bounded above by $\text{cap}_u^{\mathcal{A}}(S)$:

$$|\Phi_u(S)| \leq \text{cap}_u^{\mathcal{A}}(S).$$

4.2 Sequences

Remark 4.2.1. We present idealized sequences giving the principle ideas behind our toolbox. We introduce these idealized sequences so as to give some intuition regarding the purpose of the concrete sequences presented later on.

The actual sequences presented have weaker guarantees than the ones shown here. Still, these weaker guarantees suffice to prove the lower bounds.

Remark 4.2.2. These sequences only issue events to vertices given to them as arguments. Thus, they do not change the unfairness of any other vertex.

4.2.1 Pushing

This is a sequence on which several of the sequences presented later are based upon. This sequence transfers or "pushes" potential from one set to another.

Properties of the idealized sequence $\text{push}_u^{\mathcal{A}}(S_1, S_2)$:

Given an algorithm $\mathcal{A}$, S_1, S_2 subsets of V, $u \in \mathbb{Z}$, the idealized sequence $\text{push}_u^{\mathcal{A}}(S_1, S_2)$ has the following properties:

Let $\Phi^1_{\text{init}} = \Phi_u(S_1), \Phi^2_{\text{init}} = \Phi_u(S_2)$ be the potentials before the algorithm $\mathcal{A}$ processes $\text{push}^{\mathcal{A}}_u(S_1, S_2)$. Similarly, let $\Phi^1_{\text{end}} = \Phi_u(S_1), \Phi^2_{\text{end}} = \Phi_u(S_2)$ be the potentials after $\mathcal{A}$ processes the sequence $\text{push}^{\mathcal{A}}_u(S_1, S_2)$. If $|\Phi^1_{\text{init}} + \Phi^2_{\text{init}}| \leq \text{cap}^{\mathcal{A}}_u(S_2)$ then

$$\Phi^2_{\text{end}} = \Phi^1_{\text{init}} + \Phi^2_{\text{init}}.$$

Remark 4.2.3. From Observation 4.1.2 this implies that $\Phi^1_{\text{end}} = 0$.

Remark 4.2.4. The guarantees of $\text{push}^{\mathcal{A}}_u(S_1, S_2)$ depend on the choice of u, for a different value the guarantees won't necessarily hold.

4.2.2 Normalizing

This sequence "normalizes" the unfairness of the vertices in $S \subseteq V$ and makes them equal.

Property of the idealized sequence $\text{normalize}^{\mathcal{A}}_u(S)$:

Let $S \subseteq V$ such that $\Phi_u(S) = 0$ for some $u \in \mathbb{Z}$. Then after the algorithm $\mathcal{A}$ processes $\text{normalize}^{\mathcal{A}}_u(S)$, for every $v, w \in S$

$$u(v) = u(w) = u.$$

4.2.3 Distillation

This sequence concentrates or "distills" the potential of a set S into a subset of S. For this to be defined, the vertices of S need to be well ordered. E.g. $S = \{v_1, v_2, \ldots, v_m\}$ and $v_i < v_j$ if $i < j$.

Definition 4.2.5 ($\text{distill}^{\mathcal{A}}_u(S, k)$)**.** Given k pairwise disjoint subsets of S

$$S_1 = (v_1, v_2, \ldots, v_k), S_2 = (v_{k+1}, v_{k+2}, \ldots, v_{2k}), \ldots, S_{m/k}(v_{m-k+1}, v_{m-k+2}, \ldots v_m).$$

Let $\mathcal{A}$ be an algorithm. Define the ℓ'th push subsequence as

$$\text{push}^{\ell} = \text{push}_u^{\mathcal{A}}(S_\ell, \bigcup_{i=\ell+1}^{m/k} S_i).$$

Define $\text{distill}_u^{\mathcal{A}}(S,k)$ as the concatenation of $m/k-1$ push sequences

$$\text{distill}_u^{\mathcal{A}}(S,k) = \text{push}^1 \| \text{push}^2 \| \cdots \| \text{push}^{m/k-1}.$$

Remark 4.2.6. From the guarantees of $\text{push}_u(S_1, S_2)$, subsequent to algorithm $\mathcal{A}$ processing $\text{distill}_u^{\mathcal{A}}(S,k)$ and for any $1 \leq \ell \leq m/k$, if $|\Phi_u(S)| \leq \text{cap}_u^{\mathcal{A}}(\bigcup_{i=\ell+1}^{m/k} S_i)$ then

$$\Phi_u(\bigcup_{i=\ell+1}^{m/k} S_i) = \Phi_u(S).$$

4.2.4 Accumulation

The final sequence in our toolbox. This sequence repeatedly generates unfairness in a subset and pushes this unfairness into another subset.

First we look at a subsequence that generates the unfairness, $\text{gen}_u^{\mathcal{A}}(G_1, G_2)$.

Properties of the idealized sequence $\text{gen}_u^{\mathcal{A}}(G_1, G_2)$:

Let $\mathcal{A}$ be an algorithm, $G_1, G_2 \subset V$ and let $\Phi_1 = \Phi_u(G_1)$ be the potential before $\mathcal{A}$ processes $\text{gen}_u^{\mathcal{A}}(G_1, G_2)$ and let $\Phi_1' = \Phi_u(G_1)$ be the potential after $\mathcal{A}$ processes $\text{gen}_u^{\mathcal{A}}(G_1, G_2)$. Then $\text{gen}_u^{\mathcal{A}}(G_1, G_2)$ is a sequence such that Φ_1', Φ_1 are independent, *i.e.*, for all $k \in \mathbb{Z}$, $P(\Phi_1' = k | \Phi_1) = P(\Phi_1' = k)$ and there is a constant probability that $\Phi_1 \neq 0$.

The "accumulation" sequence uses both gen() and push() as subsequences.

Definition 4.2.7 ($\text{acc}_u^{\mathcal{A}}(G_1, G_2, A, \alpha)$)**.** Let $\mathcal{A}$ be an algorithm, $G_1, G_2, A \subset V$ be pairwise disjoint and let $\alpha \in \mathbb{N}$. Define

$$\text{acc}_u^{\mathcal{A}}(G_1, G_2, A, \alpha) = \left(\text{gen}_u^{\mathcal{A}}(G_1, G_2) \| \text{push}_u^{\mathcal{A}}(G_1, A)\right)^{\alpha},$$

where α is the number of repetitions of the subsequence $\text{gen}_u^{\mathcal{A}}(G_1, G_2) \| \text{push}_u^{\mathcal{A}}(G_1, A)$. Thus G_1, G_2 are "generator" sets and A is an "accumulator" set.

Intuitively, the sequence $\text{acc}_u^{\mathcal{A}}(G_1, G_2, A, \alpha)$ generates unfairness in G_1 and pushes it into A an α number of times. So the potential $\Phi_u(A)$ does a (biased or unbiased) random walk bounded by $-\text{cap}_u^{\mathcal{A}}(A)$ and $\text{cap}_u^{\mathcal{A}}(A)$. Thus, after many steps, we expect that $|\Phi_u(A)| = \Omega(\text{cap}_u^{\mathcal{A}}(A))$.

V Lower Bound of $\Omega(\sqrt{n})$ for the *Star Algorithm* on the Star

Consider the graph consisting of the star with $2n+2$ leaves (*i.e.*, all events are to pairs (r, v) where r is the root vertex and v is one of the leaves). Let the leaves be $V = \{1, 2, \ldots, 2n+2\}$.

The *Star Algorithm* is the following:

1. Every leaf $1 \leq v \leq 2n+2$ has a counter $x_v \in \{-1, 0, 1\}$. Let S_0, S_1, and S_{-1} be random pairwise disjoint sets of sizes $|S_0| = n+1, |S_1| = (n+1)/2, |S_{-1}| = (n+1)/2$. Initially, set $x_v = 0$ for all $v \in S_0$, set $x_v = 1$ for all $v \in S_1$, and set $x_v = -1$ for all $v \in S_{-1}$. The root r maintains a counter $x_r = -\sum_{v \in V} x_v$, which is initially equal to zero.

2. When a random event (r, v) arrives, if $x_v \neq 0$ then the algorithm orients the edge so that $x_v = 0$. If $x_v = 0$ and $x_r \neq 0$ then the algorithm orients the edge so that $|x_r|$ decreases. If $x_v = x_r = 0$ then the choice is random.

During the analysis we assume that $u(v) = x_v$, *i.e.*, the unfairness of a leaf is the value of its counter. Note that this does not change an asymptotic bound by much because $|u(v) - x_v| \leq 1$.

Remark 5.0.8. In the paper [6] we use the *Star Algorithm* to prove an upper bound on the unfairness of uniform stochastic sequences derived from the star graph or bounded genus graphs. For the upper bound it suffices that the sets S_0, S_1, S_{-1} be initialized deterministically. We note that the upper bound also holds if the initialization were random.

Contrawise, a lower bound for oblivious adversaries with deterministic initialization is trivially $\Omega(n)$, the real question arises when the S_j sets are initialized randomly. We show a non-trivial lower bound on the expected unfairness under various initialization scenarios.

We prove the following theorem.

Theorem 5.0.9. *Consider the graph consisting of the star with n leaves and root r. Then there exists a sequence of events σ, such that $E[|u(r)|] = \Omega(n)$ after σ is processed by the* Star Algorithm.

Remark 5.0.10. This lower bound sequence also works if the initialization of the counters is independent and identically distributed. I.e, if independently for all $v \in V$, $P(x_v = -1) = P(x_v = 1) = 1/4$ and $P(x_v = 0) = 1/2$.

5.1 Sequences

In this section we define concrete sequences that have similar properties to the ideal ones presented in Section 4.

5.1.1 Preliminaries

In this section we consider $\Phi_u(S)$ where $u = 0$ and S is a set of leaves. Henceforth, for $S \subseteq V$ we use the notation

$$\Phi(S) = \Phi_0(S) = \sum_{v \in S} u(v).$$

We also omit the subscript u for the sequences we define, and we omit the superscript $\mathcal{A}$ as it is implicitly assumed to be the *Star Algorithm*.

Let $\mathrm{NZ}(S) = |\{v \in S | u(v) \neq 0\}|$ be the number of leaves in S with non-zero unfairness. Note that $\mathrm{NZ}(S) = |\Phi(S)|$ iff either for all $v \in S$ $u(v) \in \{0, 1\}$ or for all $v \in S$ $u(v) \in \{-1, 0\}$.

Remark 5.1.1. In Section 4.1 we defined a notion of "potential capacity" for a set $S \subseteq V$. For the *Star Algorithm*, this notion is best approximated by $\text{cap}(S) = \text{NZ}(S)$.

5.1.2 Pushing

The sequence presented here is similar to the one presented in Section 4.2.1, albeit with slightly different guarantees.

Let $v', v'' \in V$ be distinct leaves and $S = \{v_1, v_2, v_3, \ldots, v_m\}$ a set of m leaves such that $\{v', v''\}$ and S are disjoint and m is even.

The sequence $\text{push}(\{v', v''\}, S)$ is composed of four subsequences

$$\begin{aligned}
\text{push}^{s1}(\{v', v''\}, S) &= (r, v'), (r, v''), \\
\text{push}^{d1}(\{v', v''\}, S) &= (r, v_m), (r, v_m), (r, v_{m-1}), (r, v_{m-1}), \ldots, (r, v_1), (r, v_1), \\
&\quad (r, v''), (r, v''), (r, v'), (r, v'), \\
\text{push}^{s2}(\{v', v''\}, S) &= (r, v'), (r, v''), \\
\text{push}^{d2}(\{v', v''\}, S) &= (r, v_m), (r, v_m), (r, v_{m-1}), (r, v_{m-1}), \ldots, (r, v_1), (r, v_1), \\
&\quad (r, v''), (r, v''), (r, v'), (r, v').
\end{aligned}$$

I.e,

$$\begin{aligned}
\text{push}(\{v', v''\}, S) = \text{push}^{s1}(\{v', v''\}, S) \| \text{push}^{d1}(\{v', v''\}, S) \| \\
\text{push}^{s2}(\{v', v''\}, S) \| \text{push}^{d2}(\{v', v''\}, S).
\end{aligned}$$

Remark 5.1.2. The subsequences $\text{push}^{s1}(\{v', v''\}, S)$ and $\text{push}^{s2}(\{v', v''\}, S)$ are identical as are $\text{push}^{d1}(\{v', v''\}, S)$ and $\text{push}^{d2}(\{v', v''\}, S)$. The differentiation between them is helpful in the analysis.

Let $\Phi_{\text{init}}(S) = \Phi(S)$, $\Phi_{\text{init}}(\{v', v''\}) = \Phi(\{v', v''\})$ be the potentials prior to processing the sequence $\text{push}(\{v', v''\}, S)$, $\Phi_{\text{end}}(S) = \Phi(S)$, $\Phi_{\text{end}}(\{v', v''\}) = \Phi(\{v', v''\})$ be the potentials subsequent to processing the sequence $\text{push}(\{v', v''\}, S)$.

The following lemma states the effect of push$(\{v', v''\}, S)$ on the potential of S.

Lemma 5.1.3. *If* $|\Phi_{\text{init}}(S) + \Phi_{\text{init}}(\{v', v''\})| \leq \text{NZ}(S)$ *then* $E[\Phi_{\text{end}}(S)] = \Phi_{\text{init}}(S) + \Phi_{\text{init}}(\{v', v''\})$.

Remark 5.1.4. This is the analog to the idealized case where $|\Phi_{\text{init}}(S)+\Phi_{\text{init}}(\{v', v''\})| \leq \text{cap}(S) \Rightarrow \Phi_{\text{end}}(S) = \Phi_{\text{init}}(S) + \Phi(\{v', v''\})$, *i.e.*, not in expectation.

Example 5.1.5. To better clarify how this sequence works, we provide an example. Let G be star with 4 vertices, $V = \{1, 2, 3, 4\}$.

We look at a possible processing of the event sequence push$(\{1, 2\}, \{3, 4\})$. This translates into the requests

- push$^{s1}(\{1, 2\}, \{3, 4\}) = (r, 1), (r, 2)$,
- push$^{d1}(\{1, 2\}, \{3, 4\}) = (r, 4), (r, 4), (r, 3), (r, 3), (r, 2), (r, 2), (r, 1), (r, 1)$,
- push$^{s2}(\{1, 2\}, \{3, 4\}) = (r, 1), (r, 2)$,
- push$^{d2}(\{1, 2\}, \{3, 4\}) = (r, 4), (r, 4), (r, 3), (r, 3), (r, 2), (r, 2), (r, 1), (r, 1)$.

Assume that the initial state is as given in Figure 5.1.

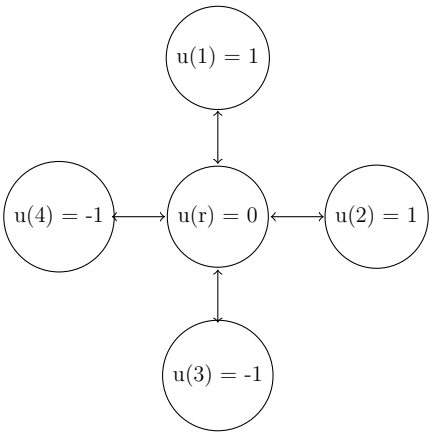

Figure 5.1: The initial state before processing push$(\{1, 2\}, \{3, 4\})$**.**

Look at the state transitions subsequent to processing push$^{s1}(\{1, 2\}, \{3, 4\})$. After processing $(r, 1)$ the only possibility is $u(r) = 1$ and $u(1) = 0$. So after processing $(r, 2)$ the state is $u(r) = 2$ and $u(2) = 0$.

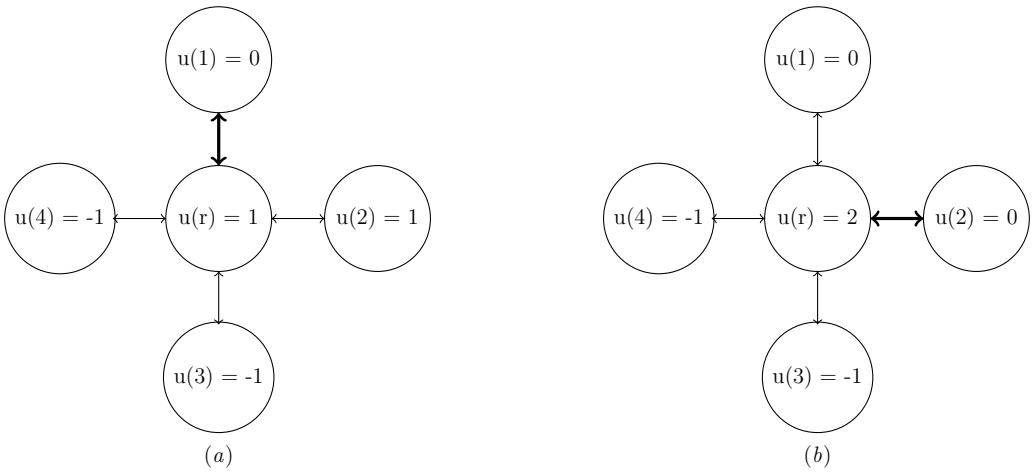

Figure 5.2: The state after: (a) processing event $(r, 1)$; (b) processing event $(r, 2)$.

Now look at the effect of processing push$^{d1}(\{1, 2\}, \{3, 4\})$. After processing the events $(r, 4), (r, 4)$, the transitions in state are $u(r) = 0$ and $u(4) = -1$. After processing the remainder of push$^{d1}(\{1, 2\}, \{3, 4\})$, which is $(r, 3), (r, 3)$, the state stays the same.

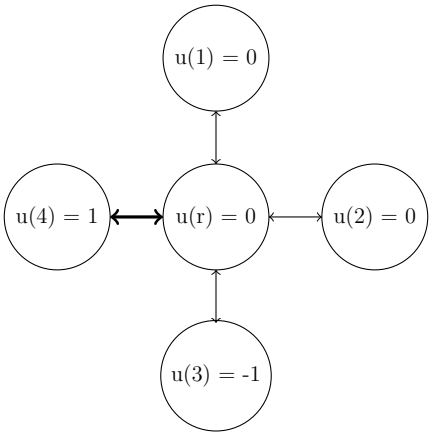

Figure 5.3: The state after processing events $(r, 4), (r, 4)$.

Subsequent to processing push$^{s2}(\{1, 2\}, \{3, 4\})$, there are two possible states: one with $u(1) = 1$, $u(2) = -1$, and $u(r) = 0$; another with $u(1) = -1$, $u(2) = 1$, and $u(r) = 0$. For the rest of this example assume that the first possibility occurred.

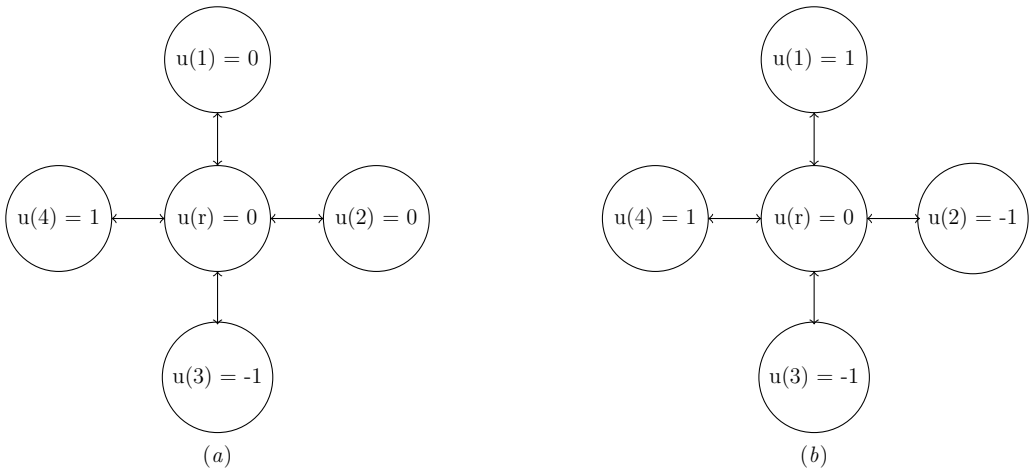

Figure 5.4: The state after: (a) processing subsequence $\text{push}^{d1}(\{1,2\},\{3,4\})$**; (b) processing subsequence** $\text{push}^{s2}(\{1,2\},\{3,4\})$**.**

The unfairness of the root is 0, so processing $\text{push}^{d2}(\{1,2\},\{3,4\})$ has no effect.

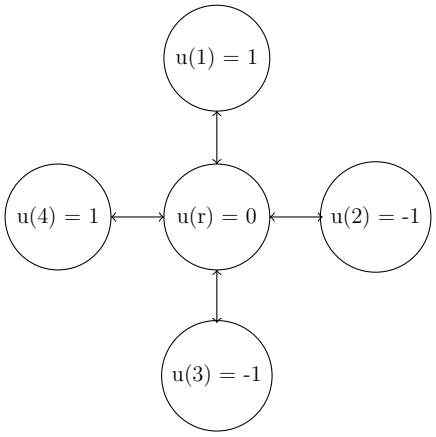

Figure 5.5: The state after processing the whole sequence $\text{push}(\{1,2\},\{3,4\})$**.**

The effect of $\text{push}(\{1,2\},\{3,4\})$ was to move the potential of the set $\{1,2\}$ to the set $\{3,4\}$.

5.1.3 Distillation

This is defined the same as in Section 4.2.3. Let $S = \{v_1, v_2, v_3, \ldots, v_m\}$ be a set of m leaves such that m is even. As we assume that $k = 2$, we use the simplified notation $\text{distill}(S)$.

Let

$$\begin{aligned}\text{distill}(S) = \text{push}(\{v_1, v_2\}, \{v_3, \ldots, v_m\}) \| \text{push}(\{v_3, v_4\}, \{v_5, \ldots, v_m\}) \| \cdots \| \\ \text{push}(\{v_{m-3}, v_{m-2}\}, \{v_{m-1}, v_m\}).\end{aligned}$$

The sequence distill(S) distills the unfairness of S into the leaves with large indices. Formally we prove the following lemma.

Lemma 5.1.6. *With probability at least* $1/2$*, there exists an* ℓ' *such that after the algorithm processes the sequence* distill(S)

- $|\Phi(S_{\ell'})| \geq |\Phi(S)|$*, and,*
- $\text{NZ}(S_{\ell'}) = |\Phi(S_{\ell'})|$.

5.1.4 Accumulation

Again, this is a sequence with similar guarantees to those of the ideal sequence presented in Section 4.2.4. Let G_1 be a set of two leaves, $G_1 = \{v', v''\}$, and let $G_2 = \{g_1, g_2, \ldots, g_n\}$, and $A = \{a_1, a_2, \ldots, a_n\}$ be sets of n leaves. Assume that G_1, G_2, and A are pairwise disjoint.

First, let the generator be

$$\text{gen}(\{v', v''\}, G_2) = \text{push}(\{v', v''\}, G_2).$$

Thus $\text{acc}(G_1, G_2, A, \alpha)$ has the structure

$$\text{acc}(G_1, G_2, A, \alpha) = \big(\text{push}(\{v', v''\}, G_2) \| \text{push}(\{v', v''\}, A)\big)^{\alpha}.$$

5.1.5 The Lower Bound Sequence

Using the subsequences introduced we now show the concrete sequence that achieves the lower bound.

Split the leaves $V = \{1, 2, 3, \ldots, 2n + 2\}$ into 3 random disjoint sets:

- $G_1 = \{v', v''\}$.
- $G_2 = \{g_1, g_2, \ldots, g_n\}$.
- $A = \{a_1, a_2, \ldots, a_n\}$.

Define the event sequence $R = (r, a_j)$, $j \in \{3n/4 + 1, 3n/4 + 2, \ldots, n\}$. The lower bound is proved by running the algorithm on the sequence

$$\text{acc}(G_1, G_2, A, \alpha) \| \text{distill}(A) \| R.$$

Remark 5.1.7. We assume that n is divisible by 4, this is not necessary.

5.2 Analysis

First, we prove a general property regarding the distribution of leaves with non-zero unfairness.

Lemma 5.2.1. *Consider any set T of leaves, any t and any sequence σ of t events. Assume that the initial configuration is that a random set of 1/2 of the leaves are assigned unfairness of zero, a random 1/4 of the leaves have unfairness of -1 and the remainder are assigned unfairness 1. Then after the* Star Algorithm *processes σ*

$$P\left(\text{NZ}(T) \geq \frac{|T|}{2}\right) \geq \frac{1}{2}.$$

Proof. The probability of any pattern of zeros/non-zeros is equal to the probability of the same pattern reversed where non-zeros exchange roles with zeros. Ergo, the probability that the number of zeros exceeds 1/2 equals the probability that the number of non-zeros exceeds 1/2. □

5.2.1 Pushing

This is the sequence as defined in Section 5.1.2.

Lemma 5.2.2. *Assume that the current configuration has* $u(r) = 0$. *Then after processing the sequence* $\text{push}(\{v', v''\}, S)$, *one still has that* $u(r) = 0$.

Proof. For simpler notation we omit the parameters of $\text{push}(\{v', v''\}, S)$ and just write push. We use this simplification also for the subsequences of push.

Table 5.1 considers the effect of the sequence push^{s1} assuming $u(r) = 0$ initially. Observe that resulting state has $u(r) \in \{-2, 0, 2\}$. Table 5.2 gives the effect of a double event of the form $(r, v), (r, v)$ $(v \in \{v', v''\} \cup S)$, assuming $u(r) \in \{-2, 0, 2\}$. Combining these effects, Table 5.3 gives the effect of the sequence $\text{push}^{s1}||\text{push}^{d1}$ assuming $u(r) = 0$ initially.

Observe that all states except $(0, 1, 1)$ and $(0, -1, -1)$ lead to $u(r) = 0$, and that no state leads to $(0, 1, 1)$ nor to $(0, -1, -1)$. Thus, it follows by applying Table 5.3 twice that the sequence $\text{push}^{s1}||\text{push}^{d1}||\text{push}^{s2}||\text{push}^{d2}$ leads to $u(r) = 0$ except possibly when going through the intermediate state $(\pm 2, 0, 0)$.

It only remains to analyze the effect of the sequence $\text{push}^{s2}||\text{push}^{d2}$ on $(\pm 2, 0, 0)$. Processing push^{s2} leads to $(0, 1, 1)$ or to $(0, -1, -1)$, and push^{d2} then keep this unchanged. Thus in all cases, after processing $\text{push}(\{v', v''\}, S)$ one still has that $u(r) = 0$.

□

Lemma 5.2.3. *Assume that initially* $u(r) = 0$. *Then for* $v \in \{v', v''\} \cup S$, $u(v) = 0$ *before processing* $\text{push}(\{v', v''\}, S)$ *iff* $u(v) = 0$ *after processing* $\text{push}(\{v', v''\}, S)$.

Proof. A more careful analysis of Tables 5.1, 5.2 and 5.3 leads to the following possibilities for $\langle u(v'), u(v'')\rangle$ after $\text{push}(\{v', v''\}, S)$ has been processed: if all the leaves in S with non-zero unfairness have the same unfairness, then processing $\text{push}(\{v', v''\}, S)$ keeps everything unchanged; otherwise, the result is distributed according to Table 5.4.

$\langle u(v'), u(v'')\rangle$ before	$\langle u(v'), u(v'')\rangle$ after	$u(r)$ after	Probability
$\langle -1, -1\rangle$	$\langle 0, 0\rangle$	**-2**	1
$\langle 0, -1\rangle$	$\langle 1, 0\rangle$ or $\langle -1, 0\rangle$	**-2** or 0	1/2, 1/2
$\langle 1, 1\rangle$	$\langle 0, 0\rangle$	**2**	1
$\langle 0, 1\rangle$	$\langle -1, 0\rangle$ or $\langle 1, 0\rangle$	**2** or 0	1/2, 1/2
$\langle -1, 0\rangle$ or $\langle 1, 0\rangle$	$\langle 0, -1\rangle$ or $\langle 0, 1\rangle$	0	1
$\langle -1, 1\rangle$ or $\langle 1, -1\rangle$	$\langle 0, 0\rangle$	0	1
$\langle 0, 0\rangle$	$\langle -1, 1\rangle$ or $\langle 1, -1\rangle$	0	1/2, 1/2

Table 5.1: Effect of events (r, v'), (r, v''), assuming $u(r) = 0$ before.

$\langle u(r), u(v)\rangle$ before	$\langle u(r), u(v)\rangle$ after
$\langle -2, -1\rangle$	unchanged
$\langle -2, 0\rangle$	unchanged
$\langle \mathbf{-2}, \mathbf{1}\rangle$	$\langle \mathbf{0}, \mathbf{-1}\rangle$
$\langle 0, -1\rangle$	unchanged
$\langle 0, 0\rangle$	unchanged
$\langle 0, 1\rangle$	unchanged
$\langle \mathbf{2}, \mathbf{-1}\rangle$	$\langle \mathbf{0}, \mathbf{1}\rangle$
$\langle 2, 0\rangle$	unchanged
$\langle 2, 1\rangle$	unchanged

Table 5.2: Effect of events $(r, v), (r, v)$ where $v \in \{v', v''\} \cup S$

$\langle u(r), u(v'), u(v'')\rangle$ before	$\langle u(r), u(v'), u(v'')\rangle$ after
$\langle 0, -1, -1\rangle$	$\langle \mathbf{0}, \mathbf{0}, \mathbf{0}\rangle$ or $\langle -2, 0, 0\rangle$
$\langle 0, -1, 0\rangle$	$\langle 0, 0, -1\rangle$
$\langle 0, -1, 1\rangle$	$\langle 0, 0, 0\rangle$
$\langle 0, 0, -1\rangle$	$\langle 0, -1, 0\rangle$ or $\langle \mathbf{0}, \mathbf{1}, \mathbf{0}\rangle$
$\langle 0, 0, 0\rangle$	$\langle 0, 1, -1\rangle$ or $\langle 0, -1, 1\rangle$
$\langle 0, 0, 1\rangle$	$\langle 0, 1, 0\rangle$ or $\langle \mathbf{0}, \mathbf{-1}, \mathbf{0}\rangle$
$\langle 0, 1, -1\rangle$	$\langle 0, 0, 0\rangle$
$\langle 0, 1, 0\rangle$	$\langle 0, 0, 1\rangle$
$\langle 0, 1, 1\rangle$	$\langle \mathbf{0}, \mathbf{0}, \mathbf{0}\rangle$ or $\langle 2, 0, 0\rangle$

Table 5.3: Effect of events $\text{push}^{s1}(\{v', v''\}, S)||\text{push}^{d1}(\{v', v''\}, S)$. The bold states are the ones where there was a change of $u(v_k)$ for some $v_k \in S$.

$\langle u(v'), u(v'')\rangle$ before	$\langle u(v'), u(v'')\rangle$ after
$\langle -1, -1\rangle$	$\langle \mathbf{-1}, \mathbf{1}\rangle$ or $\langle \mathbf{1}, \mathbf{-1}\rangle$
$\langle -1, 0\rangle$	$\langle -1, 0\rangle$ or $\langle \mathbf{1}, \mathbf{0}\rangle$
$\langle -1, 1\rangle$	$\langle -1, 1\rangle$ or $\langle 1, -1\rangle$
$\langle 0, -1\rangle$	$\langle 0, -1\rangle$ or $\langle \mathbf{0}, \mathbf{1}\rangle$
$\langle 0, 0\rangle$	$\langle 0, 0\rangle$
$\langle 0, 1\rangle$	$\langle 0, 1\rangle$ or $\langle \mathbf{0}, \mathbf{-1}\rangle$
$\langle 1, -1\rangle$	$\langle 1, -1\rangle$ or $\langle -1, 1\rangle$
$\langle 1, 0\rangle$	$\langle 1, 0\rangle$ or $\langle \mathbf{-1}, \mathbf{0}\rangle$
$\langle 1, 1\rangle$	$\langle \mathbf{-1}, \mathbf{1}\rangle$ or $\langle \mathbf{1}, \mathbf{-1}\rangle$

Table 5.4: Effect of event sequence $\mathrm{push}(\{v', v''\}, S)$**, assuming** $u(r) = 0$ **before (and therefore after), and assuming that the non-zero unfairnesses of leaves in** $S \cup \{v', v''\}$ **do not all have the same sign. The bold states are the ones where there was a change of** $u(i_k)$ **for some** $i_k \in S$**. When there are two final states, they each have probability** $1/2$**.**

Looking at Table 5.4, we note that for every $v \in \{v', v''\} \cup S$, $u(v) = 0$ after processing $\mathrm{push}(\{v', v''\}, S)$ iff we already had $u(v) = 0$ before processing $\mathrm{push}(\{v', v''\}, S)$. □

Recall that $\Phi_{\text{init}}(S) = \Phi(S)$, $\Phi_{\text{init}}(\{v', v''\}) = \Phi(\{v', v''\})$ are the potentials before processing the sequence $\mathrm{push}(\{v', v''\}, S)$; that $\Phi_{\text{end}}(S) = \Phi(S)$, $\Phi_{\text{end}}(\{v', v''\}) = \Phi(\{v', v''\})$ are the potentials after processing the sequence; and that $\mathrm{NZ}(S) = |\{v \in S | u(v) \neq 0\}|$.

Lemma 5.2.4. *If* $|\Phi_{\text{init}}(S) + \Phi_{\text{init}}(\{v', v''\})| \leq \mathrm{NZ}(S)$ *then* $E[\Phi_{\text{end}}(S)] = \Phi_{\text{init}}(S) + \Phi_{\text{init}}(\{v', v''\})$.

Proof. From Lemma 5.2.2, $u(r) = 0$ before and after the sequence is processed, and since no event concerns any leaf outside $\{v', v''\} \cup S$, we have $\Phi_{\text{init}}(\{v', v''\}) + \Phi_{\text{init}}(S) = \Phi_{\text{end}}(\{v', v''\}) + \Phi_{\text{end}}(S)$.

Looking at the right column of Table 5.4, we observe that after processing $\mathrm{push}(\{v', v''\}, S)$ the value $\Phi(\{v', v''\}) \in \{-1, 0, 1\}$ with expectation 0. This is true for any initial value of $\langle u(v'), u(v'')\rangle$. Thus $E[\Phi_{\text{end}}(S)] = \Phi_{\text{init}}(S) + \Phi_{\text{init}}(\{v', v''\})$. □

5.2.2 Distillation

We repeat the definition as given in Section 5.1.3. Let $S = \{v_1, v_2, v_3, \ldots, v_m\}$ be a set of m leaves such that m is even. For $\ell \leq m$, define the tail S_ℓ of set S:

$$S_\ell = \{v_k \in S | \ell \leq k \leq m\}.$$

.

Lemma 5.2.5. *With probability at least* $1/2$*, there exists an* ℓ' *such that after the algorithm processes the sequence* distill(S)*:*

- $|\Phi(S_{\ell'})| \geq |\Phi(S)|$*,* and*,*
- $\mathrm{NZ}(S_{\ell'}) = |\Phi(S_{\ell'})|$.

Proof. For any ℓ such that $|\Phi_{\text{init}}(S) + \Phi_{\text{init}}(\{v', v''\})| \leq \mathrm{NZ}(S)$, denote Φ_ℓ as the value of $\Phi(S_\ell)$ before processing push$(\{v_\ell, v_{\ell+1}\}, S_{\ell+2})$. From Lemma 5.2.4, $E[\Phi_{\ell+2}] = \Phi_\ell$. Thus, if we look at the random variable Φ_ℓ, it is performing an unbiased random walk with self-loops.

Let ℓ' be the first ℓ where $\mathrm{NZ}(S_\ell) = |\Phi_\ell|$. Denote this point in time t'. Given that Φ_ℓ performed an unbiased random walk, with probability $1/2$, $|\Phi_{\ell'}| \geq |\Phi(S)|$. Thus at time t', with probability $1/2$

$$|\Phi(S_{\ell'})| \geq |\Phi(S)|.$$

Given that all the later events in the sequence distill(S) are for leaves in $S_{\ell'}$, both $\Phi(S_{\ell'})$ and $\mathrm{NZ}(S_{\ell'})$ are the same at time t' and after distill(S) was processed.

□

5.2.3 Accumulation

This is the sequence defined in Section 5.1.3. Let G_1 be a set of two leaves, $G_1 = \{v', v''\}$, and let $G_2 = \{g_1, g_2, \ldots, g_n\}$, and $A = \{a_1, a_2, \ldots, a_n\}$ be sets of n leaves. Assume that G_1, G_2, A are pairwise disjoint.

First, let the generator be

$$\text{gen}(\{v', v''\}, G_2) = \text{push}(\{v', v''\}, G_2).$$

Thus $\text{acc}(G_1, G_2, A, \alpha)$ has the structure

$$\text{acc}(G_1, G_2, A, \alpha) = (\text{push}(\{v', v''\}, G_2) \| \text{push}(\{v', v''\}, A))^{\alpha}. \tag{5.1}$$

Lemma 5.2.6. *With probability* $1/16$*, for a large enough* $\alpha \in \mathbb{N}$*, after the* Star Algorithm *processes the sequence* $\text{acc}(G_1, G_2, A, \alpha)$

$$|\Phi(A)| \geq \frac{\text{NZ}(A)}{2}.$$

Proof. Initially, with probability $1/2$, exactly one of $u(v')$ and $u(v'')$ is non-zero. Independently and with probability at least $1/4$, $\text{NZ}(A) < \text{NZ}(G_2) - 1$. Assume that the sets $\{v', v''\}, G_2, A$ were chosen such that both of these are true.

From Table 5.4, the $\text{push}(\{v', v''\}, G_2)$ and $\text{push}(\{v', v''\}, A)$ subsequences don't change the property that exactly one of $u(v'), u(v'')$ is non-zero nor do they change the property that $u(r) = 0$.

From $u(r) = 0$ we conclude $\Phi(A) + \Phi(G_2) + u(v') + u(v'') = 0$ and adding the fact $|u(v') + u(v'')| = 1$ we get $|\Phi(A) + \Phi(G_2)| = 1$. Given that $\Phi(A), \Phi(G_2) \in \mathbb{Z}$, they must not have the same sign, so $|\Phi(A) + \Phi(G_2)| = ||\Phi(A)| - |\Phi(G_2)||$.

We proceed by contradiction. Assume that $|\Phi(G_2)| = \text{NZ}(G_2)$. Then $|\Phi(A) + \Phi(G_2)| = ||\Phi(A)| - |\Phi(G_2)|| = ||\Phi(A)| - \text{NZ}(G_2)|$. But $|\Phi(A)| \leq NZ(A)$ and we assumed that $\text{NZ}(A) < \text{NZ}(G_2) - 1$ so $||\Phi(A)| - \text{NZ}(G_2)| = \text{NZ}(G_2) - |\Phi(A)| \geq$

$\mathrm{NZ}(G_2) - \mathrm{NZ}(A) > 1$ which is a contradiction to $|\Phi(A) + \Phi(G_2)| = 1$. Thus $|\Phi(G_2)| < \mathrm{NZ}(G_2)$.

If $|\Phi(G_2)| < NZ(G_2)$ then from Table 5.4, after processing each push$(\{v', v''\}, G_2)$ subsequence: with probability 1/2 the sum $u(v') + u(v'')$ is unchanged and with probability 1/2 it is negated. Thus, regardless of the value of $u(v') + u(v'')$ before processing push$(\{v', v''\}, G_2)$, the distribution of $u(v') + u(v'')$ before processing push$(\{v', v''\}, A)$ is:

$$P(u(v') + u(v'') = -1) = \frac{1}{2},$$
$$P(u(v') + u(v'') = 1) = \frac{1}{2}.$$

We define a Markov chain (shown in Figure 5.6) of the state of $\Phi(A)$, where a transition occurs each time a sequence push$(\{v', v''\}, A)$ is processed. The probabilities of the transitions are:

For $\Phi \notin \{-\mathrm{NZ}(A), \mathrm{NZ}(A)\}$

$$P(\Phi \to \Phi - 2) = \frac{1}{4},$$
$$P(\Phi \to \Phi) = \frac{1}{2},$$
$$P(\Phi \to \Phi + 2) = \frac{1}{4}.$$

For $\Phi = -\mathrm{NZ}(A)$

$$P(\Phi \to \Phi) = \frac{3}{4},$$
$$P(\Phi \to \Phi + 2) = \frac{1}{4}.$$

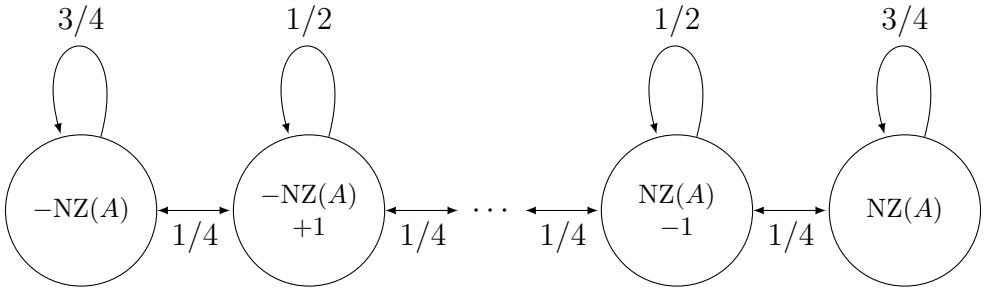

Figure 5.6: The Markov chain for $\Phi(A)$. The node labels are the potential $\Phi(A)$ at that state, the edge labels are the transition probabilities.

For $\Phi = \mathrm{NZ}(A)$

$$P(\Phi \to \Phi - 2) = \frac{1}{4},$$
$$P(\Phi \to \Phi) = \frac{3}{4}.$$

This is an ergodic, regular Markov chain. Thus the limit of the distribution is a stationary distribution. Moreover, the transition matrix is symmetric, thus the stationary distribution is uniform. $\Phi(A)$ is bounded between $-\mathrm{NZ}(A)$ and $\mathrm{NZ}(A)$. Then for a large enough α, with probability $\geq 1/2$, after the *Star Algorithm* processes $\mathrm{acc}(G_1, G_2, A, \alpha)$

$$|\Phi(A)| \geq \frac{\mathrm{NZ}(A)}{2}.$$

□

Corollary 5.2.7. *With probability at least* $1/32$*, after the* Star Algorithm *processes the sequence* $\mathrm{acc}(G_1, G_2, A, \alpha)$

$$|\Phi(A)| \geq \frac{n}{4}.$$

Proof. From Lemma 5.2.6

$$P\left(|\Phi(A)| \geq \frac{\mathrm{NZ}(A)}{2}\right) \geq \frac{1}{16}.$$

Applying Lemma 5.2.1 to A after the sequence $\mathrm{acc}(G_1, G_2, A, \alpha)$ has been processed

$$P\Big(\mathrm{NZ}(A) \geq \frac{|A|}{2}\Big) = P\Big(\mathrm{NZ}(A) \geq \frac{n}{2}\Big) \geq \frac{1}{2}.$$

These two are independent, thus

$$\begin{aligned} P\Big(|\Phi(A)| \geq \frac{n}{4}\Big) &\geq P\Big(|\Phi(A)| \geq \frac{\mathrm{NZ}(A)}{2} \bigwedge \mathrm{NZ}(A) \geq \frac{n}{2}\Big) \\ &= P\Big(|\Phi(A)| \geq \frac{\mathrm{NZ}(A)}{2}\Big) \cdot P\Big(\mathrm{NZ}(A) \geq \frac{n}{2}\Big) \\ &\geq \frac{1}{32}. \end{aligned}$$

□

5.2.4 The Lower Bound Sequence

Recall the lower bound sequence presented in Section 5.1.5,

$$\mathrm{acc}(G_1, G_2, A, \alpha) \| \mathrm{distill}(A) \| R.$$

Let $A' = \{a_j | 3n/4 + 1 \leq j \leq n\}$ be the last $n/4$ indices in A.

Lemma 5.2.8. *After the* Star Algorithm *processes the sequence* $\mathrm{acc}(G_1, G_2, A, \alpha) \| \mathrm{distill}(A)$*, we have*

$$P\Big(|\Phi(A')| \geq \frac{n}{8}\Big) \geq \frac{1}{128}.$$

Proof. Applying Lemma 5.2.1 to A' after the sequence $\mathrm{acc}(G_1, G_2, A, \alpha) \| \mathrm{distill}(A)$ has been processed

$$P\Big(\mathrm{NZ}(A') \geq \frac{|A'|}{2}\Big) = P\Big(\mathrm{NZ}(A') \geq \frac{n}{8}\Big) \geq \frac{1}{2}. \tag{5.2}$$

From Corollary 5.2.7

$$P\Big(|\Phi(A)| \geq \frac{n}{4}\Big) \geq \frac{1}{32}. \tag{5.3}$$

The probability that the Equation 5.2 holds is not independent from the probability that Equation 5.3 holds, but they are positively correlated. Thus,

$$P\Big(\Big(\mathrm{NZ}(A') \geq \frac{n}{8}\Big) \wedge \Big(|\Phi(A)| \geq \frac{n}{4}\Big)\Big) \geq \frac{1}{64}. \tag{5.4}$$

From Lemma 5.2.5, with probability 1/2, after distill(A) is processed there exists an ℓ' such that

$$|\Phi(A_{\ell'})| \geq |\Phi(A)|, \text{ and, } \mathrm{NZ}(A_{\ell'}) = |\Phi(A_{\ell'})|. \tag{5.5}$$

These events in Equations 5.4 and 5.5 are independent, so they occur simultaneously with probability $\geq 1/128$.

From $|\Phi(A)| \geq n/4$, we obtain that $\ell' \leq 3n/4$ because $|\Phi(A_{\ell'})| \leq |A_{\ell'}|$ and $|\Phi(A_{\ell'})| \geq |\Phi(A)|$. Thus $A' \subseteq A_{\ell'}$ so $\mathrm{NZ}(A') = |\Phi(A')|$.

From $\mathrm{NZ}(A') \geq n/8$, we get that $|\Phi(A')| \geq n/8$.

□

Theorem 5.2.9. *The sequence* acc$(G_1, G_2, A, \alpha)\|$distill$(A)\|R$ *achieves an expected unfairness of* $\Omega(\sqrt{n})$ *on the* Star Algorithm *when run on a star graph with* n *leaves.*

Proof. From Lemma 5.2.8, after processing acc$(G_1, G_2, A, \alpha)\|$distill(A)

$$P\Big(|\Phi(A')| \geq \frac{n}{8}\Big) \geq \frac{1}{128}.$$

Assume WLOG that $\Phi(A') \geq n/8$. Then at this point in time, (prior to processing R), at least $n/8$ of the leaves in A' have unfairness 1 and at most $n/8$ have unfairness 0.

The effect of the events R are similar to a random walk for $u(r)$. Each time leaf a with unfairness 1 is requested, the value $u(r)$ increases by 1. If $u(r) > 0$ then each time a leaf with unfairness 0 is requested, then $u(r)$ decreases by 1. If $u(r) \leq 0$ then requesting a leaf with unfairness 0 might increase $u(r)$. Thus, a random walk

is a lower bound on the unfairness of $u(r)$. A random walk of length $n/8$ has an expected value of $\Omega(\sqrt{n})$. So, after processing the events R the expected value of $u(r)$ is $\Omega(\sqrt{n})$.

This occurs with probability greater than $1/128$, so the expected unfairness of r after processing $\text{acc}(G_1, G_2, A, \alpha) \| \text{distill}(A) \| R$ is

$$\Omega(\sqrt{n}).$$

□

VI Lower Bound of $\Omega(\log n)$ for *Randomized Global Greedy* on the Clique

In this section we show a lower bound of $\Omega(\log n)$ on the *Randomized Global Greedy* (which we refer to as *RGG*) algorithm when applied to a clique with n vertices. *I.e.*, all events are pairs (v, w) where $v \neq w$ are vertices. We show this lower bound for event sequences generated by oblivious adversaries.

Given that the next even is (v, w), *Randomized Global Greedy* does the following:

- If $u(v) > u(w)$ then set w to drive. This decreases $u(v)$ and increases $u(w)$.
- If $u(v) < u(w)$ then set v to drive. This increases $u(v)$ and decreases $u(w)$.
- If $u(v) = u(w)$ then set v to drive with probability $1/2$ and set w to drive with probability $1/2$.

6.1 Sequences

In this section we present sequences that are similar to the ideal ones presented in Section 4.

Let V be the set of vertices, $V = \{1, 2, \ldots, n\}$, and let $u(v)$ be the unfairness of a vertex $v \in V$. Define the potential function for $S \subseteq V$ and $u \in \mathbb{Z}$ as

$$\Phi_u(S) = \sum_{v \in S} (u(v) - u).$$

We omit the superscript $\mathcal{A}$ as it is implicitly assumed to be *Randomized Global Greedy*.

Remark 6.1.1. In Section 4.1 we defined a notion of "potential capacity" for a set $S \subseteq V$. In the case of *Randomized Global Greedy*, for any $u \in \mathbb{Z}$, and for any $\emptyset \neq S \subseteq V$ the value $\Phi_u(S)$ is unbounded.

Let $\mathcal{F}$ be the family of sequences defined later in this section. Let $\sigma \in \mathcal{F}$ and let $S_\sigma = \{v \in V | (v, w) \in \sigma\}$ be the set of vertices effected by σ. All the sequences defined keep the invariant: for any $v \in S_\sigma$ if $u(v) \in \{u-1, u, u+1\}$ prior to RGG processing σ then $u(v) \in \{u-1, u, u+1\}$ subsequent to RGG processing σ.

Thus, we define the potential capacity for family $\mathcal{F}$ to be

$$\text{cap}_u^{\mathcal{F}}(S) = |S|.$$

6.1.1 Pushing

This is a concrete example of a push() sequence as presented in Section 4.2.1. Let $v', v'', r \in V$ be distinct vertices and let $S = \{v_1, v_2, v_3, \ldots, v_m\}$ be a set of m vertices such that $v', v'', r \notin S$. In addition to the parameters specified in Section 4.2.1, this sequence takes as a parameter an additional vertex $r \in V$. The vertex r is a "virtual root". Also, let $u \in \mathbb{Z}$ be the "base unfairness"

Let $\alpha \in \mathbb{N}$ be a large number, as a function of $|V| = n$. This value needs to be large so that the pushing property occurs with high probability.

The sequence $\text{push}_u(\{v', v''\}, S, r)$ is composed of three subsequences

$$\begin{aligned}
\text{push}_u^{s1}(\{v', v''\}, S, r) &= (r, v'), (r, v''), \\
\text{push}_u^{d}(\{v', v''\}, S, r) &= (r, v_m), (r, v_m), (r, v_{m-1}), (r, v_{m-1}), \ldots, (r, v_1), (r, v_1), \\
\text{push}_u^{s2}(\{v', v''\}, S, r) &= (r, v''), (r, v').
\end{aligned}$$

I.e.,

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

Remark 6.1.2. Although this sequence is similar to the push() sequence used in the lower bound proof for *Star Algorithm*, there are some differences:

1. There is only one "double event" subsequence, $\text{push}_u^d(\{v', v''\}, S, r)$.
2. The events (r, v'), (r, v') and (r, v''), (r, v'') are not issued by $\text{push}_u^d(\{v', v''\}, S, r)$.
3. The second "single event" subsequence $(\text{push}_u^{s2}(\{v', v''\}, S, r))$ is in reversed order compared with the first one $(\text{push}_u^{s1}(\{v', v''\}, S, r))$. Thus, unlike in the *Star Algorithm*'s sequence, they are not equal.
4. The subsequences are repeated many, $\alpha \in \mathbb{N}$, times and not only once.

Remark 6.1.3. The number of repetitions α is not a parameter of $\text{push}_u(\{v', v''\}, S, r)$ because it is a function of $|V| = n$ and identical for each $\text{push}_u()$ sequence regardless of its parameters.

We only analyze the sequence $\text{push}_u(\{v', v''\}, S, r)$ under the assumption that predicates (a), (b), and (c) hold, defined as follows: (a) For some $u \in \mathbb{Z}$, $u(r) = u$, (b) That $u(v'), u(v'') \in \{u - 1, u + 1\}$, and (c) For all $v \in S$, $u(v) \in \{u - 1, u + 1\}$.

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. We prove the following lemma.

Lemma 6.1.4. *If* $|u(v') + u(v'') + \Phi_{\text{init}}(S)| <= |S|$ *then, with high probability, after the RGG 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.*

Example 6.1.5. To better clarify how this sequence works, we provide an example. Let $|V| = 5$ vertices. Denote one of the vertices as the "virtual root" r and denote the others as $\{1, 2, 3, 4\}$. We look at the case $\alpha = 2$.

Look at a possible processing of the event sequence $\text{push}_0(\{1, 2\}, \{3, 4\}, r)$. This translates into events

- $\text{push}_0^{s1}(\{1,2\},\{3,4\},r) = (r,1),(r,2),$

- $\text{push}_0^{d}(\{1,2\},\{3,4\},r) = (r,4),(r,4),(r,3),(r,3),$

- $\text{push}_0^{s2}(\{1,2\},\{3,4\},r) = (r,2),(r,1).$

Thus

$$\text{push}_0(\{1,2\},\{3,4\},r) = ((r,1),(r,2),(r,4),(r,4),(r,3),(r,3),(r,2),(r,1))^2.$$

Assume that the initial state is as given in Figure 6.1.

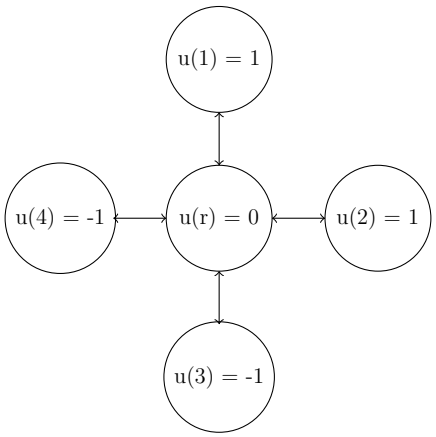

Figure 6.1: The state before processing $\text{push}_0(\{1,2\},\{3,4\},r)$**.**

Look at the state transitions when processing the sequence $\text{push}^{s1}(\{1,2\},\{3,4\})$. There is only one possibility after processing event $(r,1)$, $u(r) = 1$ and $u(1) = 0$. But, after processing $(r,2)$ there are two possible states: one with $u(r) = 0$ and $u(2) = 2$; another with $u(r) = 2$ and $u(2) = 0$. In this example, we assume that the first possible state occurred.

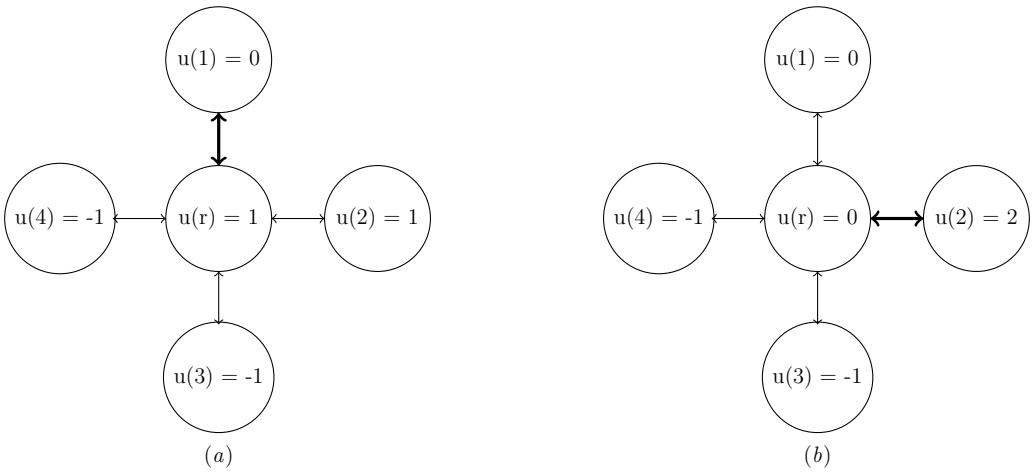

Figure 6.2: The state after: (a) processing event $(r, 1)$; (b) processing event $(r, 2)$.

Given that $u(r) = 0$, the processing of $\text{push}_0^d(\{1, 2\}, \{3, 4\}, r)$ has no effect. Thus, before processing $\text{push}_0^d(\{1, 2\}, \{3, 4\}, r)$, $u(r) = 0$, $u(1) = 0$, and $u(2) = 2$. Processing $\text{push}_0^{s2}(\{1, 2\}, \{3, 4\}, r)$ returns the state to its initial value, $u(r) = 0$, $u(1) = 1$, and $u(2) = 1$.

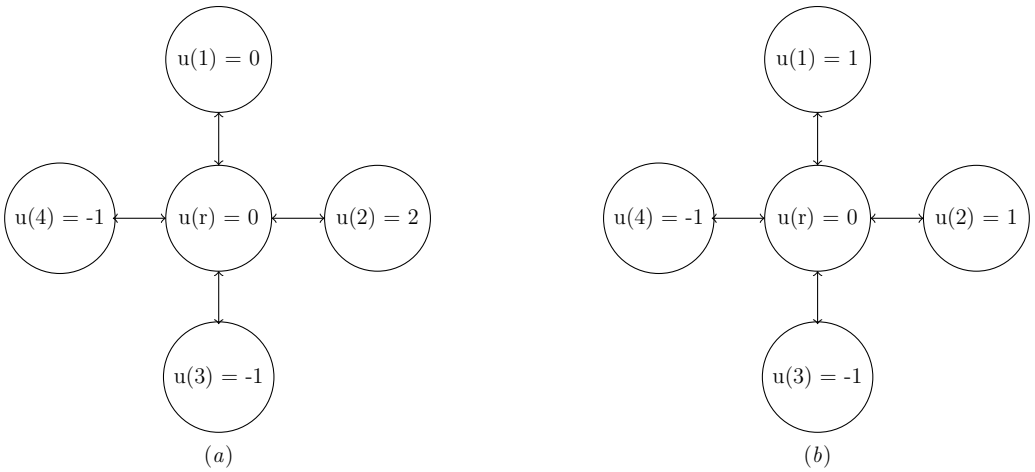

Figure 6.3: The state after: (a) processing subsequence $\text{push}_0^d(\{1, 2\}, \{3, 4\}, r)$; (b) processing subsequence $\text{push}_0^{s2}(\{1, 2\}, \{3, 4\}, r)$.

We now examine the effects of the second repetition of $\text{push}_0^{s1} \| \text{push}_0^d \| \text{push}_0^{s2}$. There is still only one possibility after processing event $(r, 1)$, $u(r) = 1$, and $u(1) = 0$. But this repetition, after processing $(r, 2)$ we assume that $u(r) = 2, u(1) = 0$, and $u(2) = 0$.

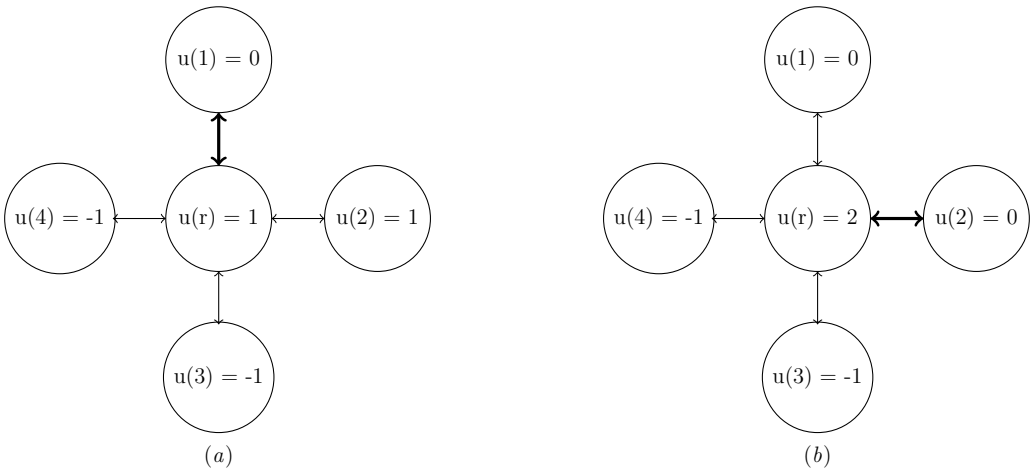

Figure 6.4: The state after: (a) processing event $(r, 1)$; (b) processing event $(r, 2)$.

Now $u(r) = 2$ and subsequent to processing $\text{push}_0^d(\{1,2\},\{3,4\},r)$, $u(4) = 1$ and $u(r) = 0$. Processing $\text{push}_0^{s2}(\{1,2\},\{3,4\},r)$ can lead to two states: the first $u(1) = -1$ and $u(2) = 1$; the second $u(1) = 1$ and $u(2) = -1$. Assume that the first occurred.

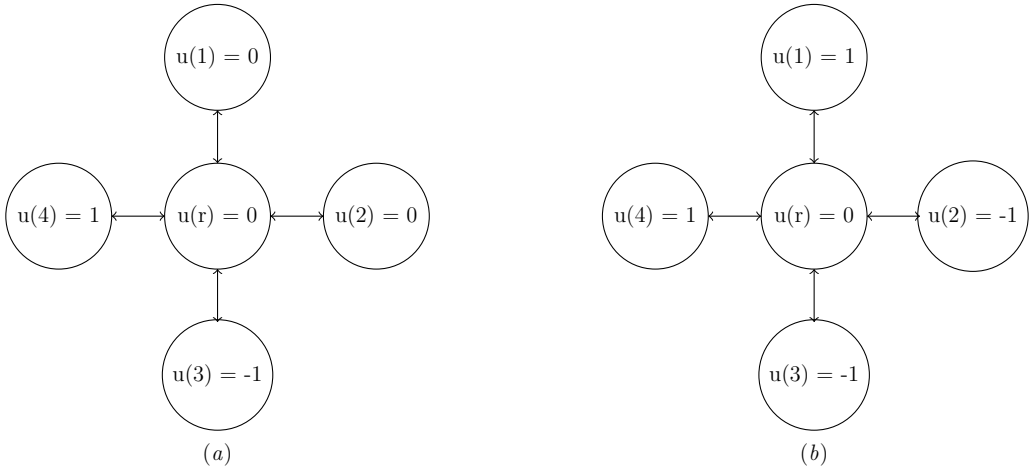

Figure 6.5: The state after: (a) processing event subsequence $\text{push}_0^d(\{1,2\},\{3,4\},r)$; (b) processing subsequence $\text{push}_0^{s2}(\{1,2\},\{3,4\},r)$.

The state shown in Figure 6.5 (b) is the final state after processing $\text{push}_0(\{1,2\},\{3,4\},r)$. The overall effect of sequence $\text{push}_0(\{1,2\},\{3,4\},r)$ was to move the potential in set $\{1,2\}$ to the set $\{3,4\}$, while keeping $u(r) = 0$.

6.1.2 Normalization

We achieve guarantees close to the ideal ones presented in 4.2.2. Let $S = \{v_1, v_2, \ldots, v_m\} \subseteq V$, $|S| = m$, and let $u \in \mathbb{Z}$. Assume that m is even.

Let $\alpha' \in \mathbb{N}$ be a large integer, as a function of $|V| = n$. Let θ_k be a sequence of random variables, each of which takes a a uniform distribution over vertices in S, furthermore they are correlated as $\theta_{2j-1} \neq \theta_{2j}$ for all j. Define $\text{normalize}_u(S)$ as follows:

$$\text{normalize}_u^1(S) = (v_1, v_2) \| (v_3, v_4) \| \cdots \| (v_{m-1}, v_m),$$
$$\text{normalize}_u^2(S) = (\theta_1, \theta_2) \| (\theta_1, \theta_2) \| (\theta_3, \theta_4) \| \| (\theta_3, \theta_4), \| \cdots \| (\theta_{\alpha'-1}, \theta_{\alpha'} \| (\theta_{\alpha'-1}, \theta_{\alpha'}),$$
$$\text{normalize}_u(S) = \text{normalize}_u^1(S) \| \text{normalize}_u^2(S).$$

We analyze this sequence only when every vertex $v \in S$ has unfairness $u(v) \in \{u-1, u+1\}$ and only when $\Phi_u(S) = 0$.

We prove the following lemma.

Lemma 6.1.6. *Let $S \subseteq V$ be a set such that $\Phi_u(S) = 0$ for some $u \in \mathbb{Z}$ and that for every vertex $v \in S$, the unfairness $u(v) \in \{u-1, u+1\}$. After the RGG algorithm processes* $\text{normalize}_u(S)$*, with high probability, for every $v \in S$ the unfairness $u(v) = u$.*

Remark 6.1.7. The number of repetitions α' is not a parameter of $\text{normalize}_u(S)$ because it is a function of $|V| = n$ and not a function of any of the parameters of $\text{normalize}_u(S)$.

6.1.3 Distillation

The definition of distill() shown here is very similar to the idealized one presented in Section 4.2.3. In addition to the parameters specified in Section 4.2.3, this sequence takes as a parameter a vertex $r \in V$. The vertex r is a "virtual root". Let $S =$

$\{v_1, v_2, v_3, \ldots, v_m\}$ be a set of m vertices such that m is even. We fix $k = 2$, and thus to simplify the notation we omit k as a parameter and write $\text{distill}_u(S, r)$.

The $\text{distill}_u(S, r)$ event sequence is

$$\begin{aligned}\text{distill}_u(S, r) = \text{push}_u(\{v_1, v_2\}, \{v_3, \ldots, v_m\}, r) \| \text{push}_u(\{v_3, v_4\}, \{v_5, \ldots, v_m\}, r) \| \\ \cdots \| \text{push}_u(\{v_{m-3}, v_{m-2}\}, \{v_{m-1}, v_m\}, r).\end{aligned}$$

Define the tail of S, S_ℓ, as $S_\ell = \{v_k \in S | k \geq \ell\}$. Formally we prove that

Lemma 6.1.8. *With high probability, after the RGG algorithm processes the sequence* $\text{distill}_u(S)$

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

Remark 6.1.9. Note that the set $S_{|S|-|\Phi_u(S)|+1}$ is the last $|\Phi_u(S)|$ vertices in S.

6.1.4 Accumulation

The sequences described herein give similar guarantees to those shown in Section 4.2.4. The new accumulation sequences are composed of two ideal generators and accumulators. Let $u \in \mathbb{Z}$ be the "base unfairness".

Choose pairwise disjoint subsets of V:

$$\begin{aligned} G_1 &= \{v_1, v_2\} \subseteq V, & G_2 &= \{v_3, v_4\} \subseteq V, \\ A_1 &= \{a_1^1, a_2^1, \ldots, a_m^1\} \subseteq V, & A_2 &= \{a_1^2, a_2^2, \ldots, a_m^2\} \subseteq V, \end{aligned}$$

and choose $r \in V \setminus (G_1 \cup G_2 \cup A_1 \cup A_2)$. Call r a *virtual root*. Assume that m is even.

Define the sequence $\mathrm{acc}_u(G_1, G_2, A_1, A_2, r, \alpha)$:

$$\begin{aligned}
\mathrm{create}(A_i) &= (a_1^i, a_2^i), (a_1^i, a_2^i), \ldots, (a_m^i - 1, a_m^i), \\
\mathrm{gen}_u(G_1, G_2) &= \mathrm{normalize}_u(\{v_1, v_2, v_3, v_4\}) \| (v_1, v_3) \| (v_2, v_4), \\
\mathrm{acc}_u^i(G_1, G_2, A_1, A_2, r) &= \mathrm{gen}_u(G_1, G_2, r) \| \mathrm{push}_u(G_1, A_1, r) \| \mathrm{push}_u(G_2, A_2, r), \\
\mathrm{acc}_u(G_1, G_2, A_1, A_2, r, \alpha) &= \mathrm{create}(A_1) \| \mathrm{create}(A_2) \| \\
&\qquad \mathrm{acc}_u^1(G_1, G_2, A_1, A_2, r) \| \cdots \| \mathrm{acc}_u^\alpha(G_1, G_2, A_1, A_2, r).
\end{aligned}$$

We only analyze this sequence under the following conditions:

- $\Phi_u(\{v_1, v_2, v_3, v_4\}) = 0$.
- For every $v \in A_1$, $u(v) = u$.
- For every $v \in A_2$, $u(v) = u$.
- $u(r) = u$.
- $\Phi_u(A_1) = -\Phi_u(A_2)$.

We prove the following lemma.

Lemma 6.1.10. *Assume the conditions above hold. After the RGG algorithm processes the sequence* $\mathrm{acc}_u(G_1, G_2, A_1, A_2, \alpha)$, $P\big((|\Phi_u(A_1)| \geq m/2) \wedge (|\Phi_u(A_2)| \geq m/2)\big) \geq 1/2$.

6.1.5 The Lower Bound Sequence

In this section we show how to use the "accumulate" sequence in order to achieve a $\Omega(\log n)$ bound for the *Randomized Global Greedy* algorithm. To do this, we define a sequence $\mathrm{iter}(S)$. Intuitively, $\mathrm{iter}(S)$ splits the vertices into 3 pairs of sets, "accumulates" each pair, and then recursively calls iter() $\Omega(\log n)$ times.

Remark 6.1.11. In this section we assume that m is divisible by 12, but this is not necessary. If $m = k \pmod{12}$ where $k \neq 0$ then redefine S to be S without k vertices.

Let there be a "base unfairness" $u \in \mathbb{Z}$ and let $S \subseteq V$, $|S| = m + 5$, be a set of vertices. Partition S into the following:

1. A virtual root r.

2. Two generator sets G_1, G_2, each of size 2.

3. Three pairs of sets (A_1^1, A_2^1),(A_1^2, A_2^2), and (A_1^3, A_2^3) where for $i \in \{1, 2, 3\}, j \in \{1, 2\}$, $|A_j^i| = m/6$.

We only analyze this sequence when, for some $u \in \mathbb{Z}$, it holds that for every $v \in S$, $u(v) = u$.

For $i \in \{1, 2, 3\}, j \in \{1, 2\}$ denote $A_j^i = \{(a_j^i)_k \mid 1 \leq k \leq m/6\}$ and $(A_j^i)' = \{(a_j^i)_k \mid m/12 + 1 \leq k \leq m/6\}$ (*i.e.*, the last $m/12$ vertices).

Define the subsequences $\text{iter}_i(S)$, for $i \in \{1, 2, 3\}$ and $\alpha \in \mathbb{N}$ as

$$\text{iter}_i(S) = \text{acc}_u(G_1, G_2, A_1^i, A_2^i, r, \alpha) \| \text{distill}(A_1^i, r) \| \text{distill}(A_2^i, r) \| \text{iter}((A_1^i)') \| \text{iter}((A_2^i)').$$

Finally, define the sequence

$$\text{iter}(S) = \text{iter}_1(S) \| \text{iter}_2(S) \| \text{iter}_3(S).$$

These recursive calls can continue until reaching a depth of $\Omega(\log n)$ because the sizes of the sets decrease exponentially, *i.e.*,

$$\frac{|(A_1^i)'|}{|S|} = \frac{|(A_2^i)'|}{|S|} = \frac{1 - 5/|S|}{12} \approx \frac{1}{12}.$$

The lower bound for RGG is achieved proven by giving the sequence iter(V) as input.

Theorem 6.1.12. *The sequence* iter(V) *achieves an expected unfairness of* $\Omega(\log n)$ *for* Randomized Global Greedy *on the clique.*

6.2 Analysis

Denote V as the set of vertices, $V = \{1, 2, \ldots, n\}$. Let $u(v)$ be the unfairness of a vertex where $v \in V$.

For $S \subseteq V$ define the potential of S as

$$\Phi_u(S) = \sum_{v \in S} (u(v) - u).$$

6.2.1 Pushing

For clarity, we repeat the definition given in Section 6.1.1. Fix the "base unfairness" to be u for some $u \in \mathbb{Z}$. The sequence $\text{push}_u(\{v', v''\}, S, r)$ has three parameters:

1. A "virtual root" r such that $u(r) = u$.

2. Two vertices, whose unfairness will be pushed, v' and v'' such that $v', v'' \in \{u - 1, u + 1\}$

3. A set S, which the unfairness will be pushed into, such that for every $v \in S$ $u(v) \in \{u - 1, u + 1\}$.

Let $S = \{v_1, v_2, v_3, \ldots, v_m\}$. Recall the definition of $\text{push}_u(\{v', v''\}, S, r)$

$$\text{push}_u(\{v', v''\}, S) = \left(\text{push}_u^{s1}(\{v', v''\}, S, r) \| \text{push}_u^{d}(\{v', v''\}, S, r) \| \text{push}_u^{s2}(\{v', v''\}, S, r)\right)^{\alpha},$$

$\langle u(v'), u(v'')\rangle$ before	$\langle u(v'), u(v'')\rangle$ after	r after	Probability
$\langle u-1, u-1\rangle$	$\langle u, u\rangle$	**u-2**	1/2
$\langle u-1, u-1\rangle$	$\langle u, u-2\rangle$	u	1/2
$\langle u-1, u+1\rangle$	$\langle u, u\rangle$	u	1
$\langle u+1, u-1\rangle$	$\langle u, u\rangle$	u	1
$\langle u+1, u+1\rangle$	$\langle u, u+2\rangle$	u	1/2
$\langle u+1, u+1\rangle$	$\langle u, u\rangle$	**u+2**	1/2

Table 6.1: Effect of events (r, v'), (r, v''), where $u(v'), u(v'') \in \{u-1, u+1\}$ and assuming $u(r) = u$ before the pair arrived.

where

$$\text{push}_u^{s1}(\{v', v''\}, S, r) = (r, v'), (r, v''),$$

$$\text{push}_u^{d}(\{v', v''\}, S, r) = (r, v_m), (r, v_m), (r, v_{m-1}), (r, v_{m-1}), \ldots, (r, v_1), (r, v_1),$$

$$\text{push}_u^{s2}(\{v', v''\}, S, r) = (r, v''), (r, v').$$

Lemma 6.2.1. *Let r be a "virtual root", $T = \{v', v''\} \cup S$. Assume that $u(r) = u$ and that for every vertex v in T, $u(v) \in \{u-1, u+1\}$. Then — processing the subsequence $\text{push}_u^{s1}(\{v', v''\}, S, r) \| \text{push}_u^{d}(\{v', v''\}, S, r) \| \text{push}_u^{s2}(\{v', v''\}, S, r)$ keeps $u(r) = u$ unchanged and $u(v) \in \{u-1, u+1\}$ still holds.*

Proof. For brevity the parameters of $\text{push}_u(\{v', v''\}, S, r)$ and its subsequences are omitted in this proof.

Initially, $u(r) = u$ and for every vertex $v \in T$ it holds that $u(v) \in \{u-1, u+1\}$. Table 6.1 considers the of subsequence push_u^{s1} and summarizes the possible values of $u(r), u(v'),$ and $u(v'')$.

Note that after processing push_u^{s1} it is possible that $u(v') \in \{u-2, u, u+2\}$ or that $u(v'') \in \{u-2, u, u+2\}$.

If after push_u^{s1}, $u(r) = u$ then from Table 6.2 the subsequence push_u^{d} does not affect either $u(r)$ or any $u(v_k)$.

$u(r)$ before	$u(v_k)$ before	$u(r)$ after	$u(v_k)$ after
u	u-1	u	u-1
u	u+1	u	u+1
u-2	u-1	u-2	u-1
u-2	**u+1**	**u**	**u-1**
u+2	**u-1**	**u**	**u+1**
u+2	u+1	u+2	u+1

Table 6.2: Effect of events, $(r, v_k), (r, v_k)$. Note that $u(r)$ and $u(v_k)$ are unchanged except in two cases: (a) if prior to $(r, v_k), (r, v_k)$ being processed, $u(r) = u - 2$ and $u(v_k) = u + 1$ or — (b) $u(r) = u + 2$ and $u(v_k) = u - 1$.

$\langle u(v'), u(v'')\rangle$ before	$u(r)$ before	$\langle u(v'), u(v'')\rangle$ after	$u(r)$ after	Probability
$\langle u, u-2\rangle$	u	$\langle u-1, u-1\rangle$	u	1
$\langle u, u\rangle$	u-2	$\langle u-1, u-1\rangle$	u	1
$\langle u, u\rangle$	u	$\langle u-1, u+1\rangle$	u	1/2
$\langle u, u\rangle$	u	$\langle u+1, u-1\rangle$	u	1/2
$\langle u, u\rangle$	u+2	$\langle u+1, u+1\rangle$	u	1
$\langle u, u+2\rangle$	u	$\langle u+1, u+1\rangle$	u	1

Table 6.3: Effect of events (r, v''), (r, v'), when $u(r) \in \{u-2, u, u+2\}$ before the pair arrived and $\langle u(v'), u(v'')\rangle$ is possible after push_u^{s1}

If after push_u^{s1}, $u(r) = u - 2$ then let k, $1 \leq k \leq m$ be the largest index such that $u(v_k) = u + 1$. After push_u^d is processed then $u(v_k) = u - 1$, $u(r) = u$, and all the other values are unchanged.

If after push_u^{s1}, $u(r) = u + 2$ then let k, $1 \leq k \leq m$ be the largest index such that $u(v_k) = u - 1$. After push_u^d is processed then $u(v_k) = u + 1$, $u(r) = u$, and all the other values are unchanged.

It might be that in the cases $u(r) = u - 2$ or $u(r) = u + 2$ there is no such k. Thus processing push_u^d does not change either $u(r)$ or any $u(v_k)$. In this case, after processing the sequence push_u^{s2} then from Table 6.3 the value of $u(r) = u$, $u(v') = u(v'')$ and $u(v'), u(v'') \in \{u - 1, u + 1\}$.

Thus, after processing push_u^{s2} both $u(r) = u$ and $u(v'), u(v'')$ are in $\{u-1, u+1\}$ so this is correct after processing the whole subsequence $\text{push}_u^{s1} \| \text{push}_u^d \| \text{push}_u^{s2}$. □

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

Lemma 6.2.2. *If* $|u(v') + u(v'') + \Phi_{\text{init}}(S)| <= |S|$ *then, with high probability, after the RGG 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. The parameters of $\text{push}_u(\{v', v''\}, S, r)$ and its subsequences are omitted in this proof.

From Lemma 6.2.1, each time before processing push_u^{s1} the unfairness of the root is u, *i.e.*, $u(r) = u$, and for every vertex $v \in S$, $u(v) \in \{u-1, u+1\}$.

Look at different possible values of $\langle u(v'), u(v'')\rangle$ before push_u^{s1} was processed and analyze the effect of processing the subsequence $\text{push}_u^{s1} \| \text{push}_u^{d} \| \text{push}_u^{s2}$.

1. If $\langle u(v'), u(v'')\rangle = \langle u-1, u-1\rangle$ then with probability 1/2, $u(r) = u-2$ after processing push_u^{s1}. We assumed $|u(v') + u(v'') + \Phi_{\text{init}}(S)| <= |S|$, so after processing push_u^{d} there exists a $w_1 \in S$ where $u(w_1)$ decreases by 2. With probability 1/2, $u(r)$ stays the same, processing push_u^{d} doesn't change the unfairness of any vertex, and after push_u^{s2}, $\langle u(v'), u(v'')\rangle = \langle u-1, u-1\rangle$.

2. If $\langle u(v'), u(v'')\rangle = \langle u+1, u+1\rangle$ then with probability 1/2, $u(r) = u+2$ after processing push_u^{s1}. We assumed $|u(v') + u(v'') + \Phi_{\text{init}}(S)| <= |S|$, so after processing push_u^{d} there exists a $w_{-1} \in S$ where $u(w_{-1})$ increases by 2. With probability 1/2, $u(r)$ stays the same, processing push_u^{d} doesn't change the unfairness of any vertex, and after push_u^{s2}, $\langle u(v'), u(v'')\rangle = \langle u+1, u+1\rangle$.

3. In all other cases, $u(r)$ doesn't change after processing push_u^{s1} so processing push_u^{d} doesn't change the unfairness of any vertex.

If $u(v') \neq u(v'')$ then after processing each σ_ℓ^{s2} sequence, the inequality $u(v') \neq u(v'')$ still holds and $u(v'), u(v'') \in \{u-1, u+1\}$. Thus $\Phi_{\text{end}}(S) = \Phi_{\text{init}}(S) + \Phi_{\text{init}}(\{v', v''\})$.

If $u(v') = u(v'')$ then with high probability after a σ_ℓ^{s1} subsequence is processed $u(r) \neq u$ and thus $u(v') \neq u(v'')$. After the corresponding σ_ℓ^{s2} is processed $\Phi_{\text{end}}(S) = \Phi_{\text{init}}(S) + \Phi_{\text{init}}(\{v', v''\})$. Given that now $u(v') \neq u(v'')$ this invariant keeps during subsequent processing of σ_ℓ^{s1} subsequences.

Thus, with high probability,

$$\Phi_{\text{end}}(S) = \Phi_{\text{init}}(S) + \Phi_{\text{init}}(\{v', v''\})$$

after processing push_u. □

Remark 6.2.3. As the above can happen with as high probability as needed, for the remainder of the paper we assume that it occurs with probability 1.

6.2.2 Distillation

Fix the "base unfairness" to be u for some $u \in \mathbb{Z}$. The sequence $\text{distill}_u(S, r)$, as defined in Section 6.1.3, has two parameters:

1. A "virtual root" r such that $u(r) = u$.

2. A set S such that for every $v \in S$ the unfairness $u(v) \in \{u - 1, u + 1\}$.

Let $S \subseteq V$, $|S| = m$ where m is even. Denote $S = \{v_1, v_2, \ldots, v_m\}$ and define the tail of S, $S_\ell = \{i_k \in S | k \geq \ell\}$.

Lemma 6.2.4. *With high probability, after the RGG algorithm processes the sequence* $\text{distill}_u(S, r)$

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

Proof. Let $\Phi_{\text{init}}(S_\ell) = \Phi_u(S_\ell)$ be the unfairness of S_ℓ prior to processing the subsequence $\text{push}_u(\{v_{\ell-2}, v_{\ell-1}\}, S_\ell, r)$, and let $\Phi_{\text{end}}(S_\ell) = \Phi_u(S_\ell)$ be the unfairness subsequent to processing the subsequence $\text{push}_u(\{v_{\ell-2}, v_{\ell-1}\}, S_\ell, r)$.

From Lemma 6.2.2, if $|u(v_{\ell-2}) + u(v_{\ell-1}) + \Phi_{\text{init}}(S_\ell)| <= |S_\ell|$ then $\Phi_{\text{end}}(S_\ell) = \Phi_{\text{init}}(S_\ell) + \Phi_{\text{init}}(\{v_{\ell-2}, v_{\ell-1}\})$. From induction

$$\Phi_{\text{end}}(S_\ell) = \Phi_u(S_1) = \Phi_u(S). \tag{6.1}$$

Let ℓ' be the first ℓ such that $|u(v_{\ell'-2}) + u(v_{\ell'-1}) + \Phi_{\text{init}}(S_\ell)| > |S'_\ell|$. So $|u(v_{\ell'-2}) + u(v_{\ell'-1}) + \Phi_{\text{init}}(S_\ell)| = |S'_\ell| + 2$ and $|\Phi_{\text{end}}(S_{\ell'-2})| = |u(v_{\ell'-2}) + u(v_{\ell'-1}) + \Phi_{\text{init}}(S_\ell)| = |S'_\ell| - 2 = |S_{\ell'-2}|$. Also, From Equation 6.1, $\Phi_{\text{end}}(S_{\ell'-2}) = \Phi_u(S)$. So $|S_{\ell'-2}| = |\Phi_u(S)|$.

The rest of the sequence only contains vertices in $S_{\ell'} \subseteq S_{\ell'-2}$, and from Lemma 6.2.1 the value of $u(r)$ stays u. So, after the RGG algorithm processes the sequence $\text{distill}_u(S, r)$, $\Phi_u(S_{\ell'-2}) = \Phi_u(S)$.

From the definition $|S_{\ell'-2}| = m - \ell' + 3$ and we have shown that $\Phi_u(S_{\ell'-2}) = \Phi_u(S)$ so $m - \ell' + 3 = \Phi_u(S)$ and $\ell' - 2 = |S| - |\Phi_u(S)| + 1$. □

Remark 6.2.5. As the above can happen with as high probability as needed, for the remainder of the paper we assume that it occurs with probability 1.

6.2.3 Normalization

Let $S = \{v_1, v_2, \ldots, v_m\} \subseteq V$. Recall the definition of $\text{normalize}_u(S)$ presented in Section 6.1.2:

$$\text{normalize}_u^1(S) = (v_1, v_2) \| (v_3, i_4) \| \cdots \| (v_{m-1}, v_m),$$
$$\text{normalize}_u^2(S) = (\theta_1, \theta_2) \| (\theta_1, \theta_2) \| (\theta_3, \theta_4) \| | (\theta_3, \theta_4), \| \cdots \| (\theta_{\alpha'-1}, \theta_{\alpha'} \| (\theta_{\alpha'-1}, \theta_{\alpha'}),$$
$$\text{normalize}_u(S) = \text{normalize}_u^1(S) \| \text{normalize}_u^2(S).$$

Lemma 6.2.6. *Let $S \subseteq V$ be a set such that $\Phi_u(S) = 0$ and that for every vertex $v \in S$ the unfairness $u(v) \in \{u-1, u+1\}$. Then after the RGG algorithm processes* $\text{normalize}_u(S)$*, with high probability, for every $v \in S$ $u(v) = u$.*

$\langle u(\theta_{2j-1}), u(\theta_{2j})\rangle$ before	$\langle u(\theta_{2j-1}), u(\theta_{2j})\rangle$ after
$\langle u-2, u-2\rangle$	$\langle u-2, u-2\rangle$
$\langle u-2, u\rangle$	$\langle u-2, u\rangle$
$\langle$**u-2,u+2**$\rangle$	$\langle$**u,u**$\rangle$
$\langle u, u\rangle$	$\langle u, u\rangle$
$\langle u, u+2\rangle$	$\langle u, u+2\rangle$
$\langle u+2, u+2\rangle$	$\langle u+2, u+2\rangle$

Table 6.4: Effect of events $(\theta_{2j-1}), (\theta_{2j}) \| (\theta_{2j-1}), (\theta_{2j})$, when $u(\theta_{2j-1}), u(\theta_{2j}) \in \{u-2, u, u+2\}$. The order of the vertices is not important, so symmetries have been omitted for brevity.

Proof. First look at the effect of normalize$^1_u(S)$. The only possible values of $\langle u(v_{2j-1}), u(v_{2j})\rangle$ before normalize$^1_u(S)$ is processed are $\langle u-1, u-1\rangle$, $\langle u-1, u+1\rangle$, $\langle u+1, u-1\rangle$, $\langle u+1, u+1\rangle$. Thus, after processing the sequence normalize$^1_u(S)$ it holds that $u(v_{2j-1}), u(v_{2j}) \in \{u-2, u, u+2\}$.

Table 6.4 shows the effect of processing $(\theta_{2j-1}), (\theta_{2j}) \| (\theta_{2j-1}), (\theta_{2j})$. Let $u(\theta_{2j-1}), u(\theta_{2j})$ be the values before the events were processed, and let $u'(\theta_{2j-1}), u'(\theta_{2j})$ be the values after the events were processed. The only cases in which $\langle u(\theta_{2j-1}), u(\theta_{2j})\rangle \neq \langle u'(\theta_{2j-1}), u'(\theta_{2j})\rangle$ and $\langle u(\theta_{2j-1}), u(\theta_{2j})\rangle \neq \langle u'(\theta_{2j}), u'(\theta_{2j-1})\rangle$ are $\langle u(\theta_{2j-1}), u(\theta_{2j})\rangle \in \{\langle u-2, u+2\rangle, \langle u+2, u-2\rangle\}$.

Define $\#_u(S)$ as the number of vertices in S with unfairness u

$$\#_u(S) = |\{v \in S | u(v) = u\}|.$$

It arises from Table 6.4 that a subsequence $(\theta_{2j-1}), (\theta_{2j}) \| (\theta_{2j-1}), (\theta_{2j})$ can increase the value of $\#_u(S)$ but that it cannot decrease its value. Given that $\Phi_u(S) = 0$, if there exists a $v' \in S$ such that $u(v') = u-2$ then there must exist a $v'' \in S$ such that $u(v'') = u+2$. Thus if $\#_u(S) \neq |S|$, with probability $\geq 1/m^2$, both of these are chosen and then $\#_u(S)$ increases by 2.

The value $\#_u(S)$ cannot decrease and for each subsequence $(\theta_{2j-1}), (\theta_{2j}) \| (\theta_{2j-1})$ processed it increases with probability $\geq 1/m^2$. Given that α' is large, with high

probability after processing $\text{normalize}_u(S)$ for every vertex v in S, $u(v) = u$. □

Remark 6.2.7. As the above can happen with as high probability as needed, for the remainder of the paper we assume that it occurs with probability 1.

6.2.4 Accumulation

Let $u \in \mathbb{Z}$ be the "base unfairness". Recall the definition given in Section 6.1.4.

1. Sets $G_1 = \{v_1, v_2\}$, $G_2 = \{v_3, v_4\}$.
2. Set $A_1 = \{a_1^1, a_2^1, \ldots, a_m^1\}$.
3. Set $A_2 = \{a_1^2, a_2^2, \ldots, a_m^2\}$.
4. Vertex r is "virtual root".

Assume that $\{r\}, A_1, A_2, G_1, G_2$ are pairwise disjoint.

Recall the definition of $\text{acc}_u(G_1, G_2, A_1, A_2, r, \alpha)$:

$$\begin{aligned}
\text{create}(A_i) &= (a_1^i, a_2^i), (a_1^i, a_2^i), \ldots, (a_m^i - 1, a_m^i), \\
\text{gen}_u(G_1, G_2) &= \text{normalize}_u(\{v_1, v_2, v_3, v_4\}) \| (v_1, v_3) \| (v_2, v_4), \\
\text{acc}_u^i(G_1, G_2, A_1, A_2, r) &= \text{gen}_u(G_1, G_2, r) \| \text{push}_u(G_1, A_1, r) \| \text{push}_u(G_2, A_2, r), \\
\text{acc}_u(G_1, G_2, A_1, A_2, r, \alpha) &= \text{create}(A_1) \| \text{create}(A_2) \| \\
&\qquad \text{acc}_u^1(G_1, G_2, A_1, A_2, r) \| \cdots \| \text{acc}_u^\alpha(G_1, G_2, A_1, A_2, r).
\end{aligned}$$

We only analyze this sequence under the following conditions:

$$\Phi_u(\{v_1, v_2, v_3, v_4\}) = 0. \tag{6.2}$$

$$\forall v \in A_1 . u(v) = u. \tag{6.3}$$

$$\forall v \in A_2 . u(v) = u. \tag{6.4}$$

$$u(r) = u. \tag{6.5}$$

We prove the following lemma.

Lemma 6.2.8. *Assume conditions 6.2 - 6.5 hold. After the RGG algorithm processes the sequence* $\text{gen}_u(G_1, G_2)$*, with high probability,*

$$\begin{aligned}
\Phi_u(\{v_1, v_2\}) &= -\Phi_u(\{v_3, v_4\}),\\
P(\Phi_u(\{v_1, v_2\}) = -2) &= P(\Phi_u(\{v_1, v_2\}) = 2) = 1/4,\\
P(\Phi_u(\{v_1, v_2\}) = 0) &= 1/2,\\
P(\Phi_u(\{v_3, v_4\}) = -2) &= P(\Phi_u(\{v_3, v_4\}) = 2) = 1/4, \text{ and,}\\
P(\Phi_u(\{v_3, v_4\}) = 0) &= 1/2.
\end{aligned}$$

Proof. We assumed $\Phi_u(\{v_1, v_2, v_3, v_4\}) = 0$. Both of the events in the sequence $(v_1, v_3) \| (v_2, v_4)$ are for vertices in $\{v_1, v_2, v_3, v_4\}$ thus after processing $\text{gen}_u(G_1, G_2)$ still $\Phi_u(\{v_1, v_2, v_3, v_4\}) = 0$.

The potential $\Phi_u(\{v_1, v_2, v_3, v_4\}) = \Phi_u(\{v_1, v_2\}) + \Phi_u(\{v_3, v_4\})$ so $\Phi_u(\{v_1, v_2\}) = -\Phi_u(\{v_3, v_4\})$.

After the RGG algorithm processes $\text{normalize}_u(\{v_1, v_2, v_3, v_4\})$, with high probability, $u(v_1) = u(v_2) = u(v_3) = u(v_4) = u$. The results of processing (v_1, v_3) and of processing (v_2, v_4) are independent so $P\big(u(v_1) = u(v_2)\big) = P\big(u(v_1) \neq u(v_2)\big) = 1/2$ and $P\big(u(v_3) = u(v_4)\big) = P\big(u(v_3) \neq u(v_4)\big) = 1/2$. Given that $u(v_1), u(v_2), u(v_3), u(v_4) \in \{u - 1, u + 1\}$ this concludes the proof. □

Lemma 6.2.9. *Assume conditions 6.2 - 6.5 hold. After the RGG algorithm processes the sequence* $\text{acc}_u(G_1, G_2, A_1, A_2, \alpha)$*,* $P\big((|\Phi_u(A_1)| \geq m/2) \wedge (|\Phi_u(A_2)| \geq m/2)\big) \geq 1/2$.

Proof. Define Φ_ℓ to be the value $\Phi_u(A_1)$ after the RGG algorithm processes $\text{acc}_u^\ell(G_1, G_2, A_1, A_2, r)$.

The first sequence processed is $\text{create}(A_1) \| \text{create}(A_2)$. After this sequence is processed, every vertex $v \in A_1 \cup A_2$ has unfairness $u - 1$ or v has unfairness $u + 1$. It is also still true that $\Phi(A_1) = \Phi(A_2) = 0$.

We assumed that initially $\Phi_u(\{v_1, v_2, v_3, v_4\}) = 0$. From Lemma 6.2.8, $\Phi_u(\{v_1, v_2\}) = -\Phi_u(\{v_3, v_4\})$ after the $\text{gen}_u(G_1, G_2)$ sequence is processed. After $\text{push}_u(G_1, A_1, r) \| \text{push}_u(G_2, A_2, r)$ is processed it is true that $\Phi(A_1) = -\Phi(A_2)$ since $u(r) = u$ and $\Phi_u(\{v_1, v_2\}) = \Phi_u(\{v_3, v_4\}) = 0$.

From induction, after $\text{acc}_u^i(G_1, G_2, A_1, A_2, r)$ is processed $\Phi_u(\{v_1, v_2, v_3, v_4\}) = 0$ and $\Phi(A_1) = -\Phi(A_2)$.

Define $\Phi_\ell = \Phi_u(A_1)$. Look at the value Φ_ℓ as a Markov chain (shown in Figure 6.6) , where transitions are between the times the algorithm processes $\text{acc}_u^\ell(G_1, G_2, A_1, A_2, r)$'s.

From Lemma 6.2.2, if $|\Phi_\ell| \neq |A_1|$, $\Phi_{\ell+1} = \Phi_\ell + \Phi_u(\{v_1, v_2\})$. From Lemma 6.2.8 the value $\Phi_u(\{v_1, v_2\})$ is distributed:

$$\begin{aligned} P(\Phi_u(\{v_1, v_2\}) = -2) &= 1/4, \\ P(\Phi_u(\{v_1, v_2\}) = 0) &= 1/2, \\ P(\Phi_u(\{v_1, v_2\}) = 2) &= 1/4. \end{aligned}$$

Note that the cases $|\Phi_\ell| = m$ are different then the other cases, in these cases no more unfairness can be pushed into A_1.

If $\Phi_\ell \notin \{-m, m\}$ (remember $|A_1| = m$) the probabilities between transitions are

$$\begin{aligned} P(\Phi \to \Phi - 2) &= \frac{1}{4}, \\ P(\Phi \to \Phi) &= \frac{1}{2}, \\ P(\Phi \to \Phi + 2) &= \frac{1}{4}. \end{aligned}$$

If $\Phi_\ell = m$ then the probabilities are

$$\begin{aligned} P(\Phi \to \Phi - 2) &= \frac{1}{4}, \\ P(\Phi \to \Phi) &= \frac{3}{4}. \end{aligned}$$

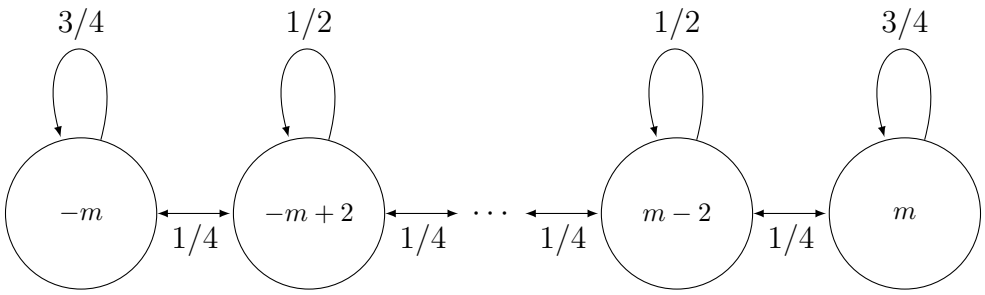

Figure 6.6: The Markov chain for Φ_ℓ. The node labels are the potential Φ_ℓ at that state, the edge labels are the transition probabilities.

If $\Phi_\ell = -m$ then the probabilities are

$$P(\Phi \to \Phi) = \frac{3}{4},$$
$$P(\Phi \to \Phi + 2) = \frac{1}{4}.$$

This is an ergodic, regular Markov chain. Thus the limit of the distribution is stationary, regardless of the starting position. Moreover, the transition matrix is symmetric, so the limit distribution is uniform. $|A_1| = m$ so, after the RGG algorithm processes $\mathrm{acc}_u(G_1, G_2, A_1, A_2, r, \alpha)$,

$$P\Big(|\Phi_u(A_1)| \geq \frac{m}{2}\Big) \geq \frac{1}{2}.$$

We proved that $\Phi_u(A_1) = -\Phi_u(A_2)$ so $|\Phi_u(A_1)| \geq m/2$ iff $|\Phi_u(A_2)| \geq m/2$.

□

6.2.5 The Lower Bound Sequence

The following is a repetition of the definition first given in Section 6.1.5. Let there be a "base unfairness" $u \in \mathbb{Z}$. Let $S \subseteq V$, $|S| = m + 5$, such that for every $v \in S$, $u(v) = u$.

Remark 6.2.10. In this section we assume that m is divisible by 12, but this is not necessary. If $m = k \pmod{12}$ where $k \neq 0$ then redefine S to be S without k

vertices.

Split S into the following

1. A virtual root r.
2. Two generator sets G_1, G_2 each of size 2.
3. Three pairs of sets (A_1^1, A_2^1),(A_1^2, A_2^2),(A_1^3, A_2^3) where for $i \in \{1,2,3\}, j \in \{1,2\}$, $|A_j^i| = m/6$.

For $i \in \{1,2,3\}, j \in \{1,2\}$ denote $A_j^i = \{(a_j^i)_k \ | 1 \leq k \leq m/6\}$ and $(A_j^i)' = \{(a_j^i)_k \ | m/12 + 1 \leq k \leq m/6\}$ (*i.e.*, the last $m/12$ vertices).

Define the subsequences $\text{iter}_i(S)$, for $i \in \{1,2,3\}$ and $\alpha \in \mathbb{N}$ a large value

$$\begin{aligned}\text{iter}_i(S) =& \text{acc}_u(G_1, G_2, A_1^i, A_2^i, r, \alpha) \| \text{distill}(A_1^i, r) \| \\ & \text{distill}(A_2^i, r) \| \text{iter}((A_1^i)') \| \text{iter}((A_2^i)').\end{aligned}$$

Define the sequence

$$\text{iter}(S) = \text{iter}_1(S) \| \text{iter}_2(S) \| \text{iter}_3(S).$$

These recursive calls can continue till a depth of $\Omega(\log n)$ because the sizes of the sets decrease exponentially, *i.e.*,

$$\frac{|(A_1^i)'|}{|S|} = \frac{|(A_2^i)'|}{|S|} = \frac{1 - 5/|S|}{12} \approx \frac{1}{12}.$$

The lower bound is proven by running the sequence $\text{iter}(V)$.

Corollary 6.2.11. *Let $i \in \{1,2,3\}, j \in \{1,2\}$. With probability $1/2$, after the* Randomized Global Greedy *algorithm processes* $\text{distill}(A_j^i, r)$

$$|\Phi_u((A_1^i)')| = |(A_1^i)'| \text{ and } |\Phi_u((A_2^i)')| = |(A_2^i)'|.$$

Proof. For $i \in \{1,2,3\}, j \in \{1,2\}$, denote $\Phi^i_j = \Phi_u(A^i_j)$ as the potential of A^i_j after the RGG algorithm processes $\text{acc}_u(G_1, G_2, A^i_1, A^i_2, r, \alpha)$. This potential is the same as the one after processing $\text{acc}_u(G_1, G_2, A^i_1, A^i_2, r, \alpha) \| \text{distill}(A^i_j, r)$.

From Lemma 6.2.9, with probability $1/2$, $|\Phi^i_1| \geq m/2$ and $|\Phi^i_2| \geq m/2$. If this occurs then from Lemma 6.2.4, after the RGG algorithm processes $\text{distill}(A^i_1, r) \| \text{distill}(A^i_2, r)$,

$$|\Phi_u((A^i_1)')| = |\Phi_u((A^i_2)')| = m/2 = |(A^i_1)'| = |(A^i_2)'|.$$

□

We show an analysis that after $\Omega(\log n)$ steps, with probability $\geq 1/2$, there exists a set $(A^i_j)'$ for $i \in \{1,2,3\}, j \in \{1,2\}$ such that all of the vertices in $(A^i_j)'$ have unfairness $\Omega(\log n)$. We show this not for the *absolute* value, the unfairness itself is large and positive. A similar analysis could be done for a negative unfairness.

Look at the $\text{iter}(V)$ calls as creating a tree. The root of the tree is V, and it has at most three children, each of the pairs of sets. These are $((A^i_1)', (A^i_2)')$ for $i \in \{1,2,3\}$. Now, $\text{iter}(A^i_j)$ is called for $j \in \{1,2\}$. This continues the tree.

We define that an edge between a parent S and a child $((A^i_1)', (A^i_2)')$ exists if $|\Phi_u((A^i_1)')| = |(A^i_1)'|$ and $|\Phi_u((A^i_2)')| = |(A^i_2)'|$. From Lemma 6.2.11 this occurs with probability $1/2$. If this occurs, exactly one of the sets $(A^i_1)', (A^i_2)'$ has $\Phi_u((A^i_j)') = |(A^i_j)'|$, *i.e.*, the potential increases.

The adversary does not know which one of the sets $(A^i_1)'$ or $(A^i_2)'$ has an increased unfairness (again, not the absolute value but the value itself). Thus, both of the sets are part of the recursive call. For the analysis, we look only at the set with increased unfairness and its children. *I.e.*, only these sets are part of the tree.

Remark 6.2.12. Let S be a set such that there is no path between S and V. In this case, it is not possible to apply Lemma 6.2.11. Randomly decide which one of the sets $(A^i_1)'$ or $(A^i_2)'$ is chosen, and define that an edge exists between S and this set with probability $1/2$.

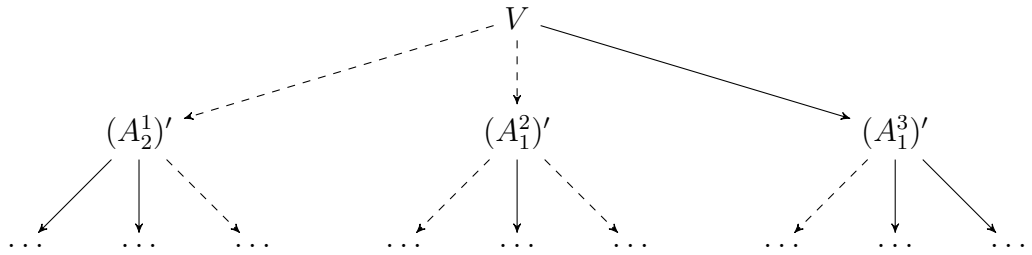

Figure 6.7: An example of a tree of sets. Solid lines are existing edges, dashed lines are edges that don't exist. The set $(A_1^i)'$, or $(A_2^i)'$ shown is the one where the unfairness of each vertex increased, or a random one if $|\Phi_u((A_1^i)')| < |(A_1^i)'|$ and $|\Phi_u((A_2^i)')| < |(A_2^i)'|$.

Lemma 6.2.13. *For all d, the probability of there being a path from the root to a vertex at depth d is at least $1/2$.*

Proof. Denote the vertex S_i^k as the i'th vertex at depth k. Note that the number of vertices at depth k is 3^k. We are interested whether the node at depth 1 has a path to a node at depth d.

Let the indicator variable X_i^k denote whether the vertex S_i^k has a path to depth d. Note that for all i, $X_i^d = 1$.

Each vertex S_i^k has 3 children, $S_{3i}^{k+1}, S_{3i+1}^{k+1}, S_{3i+2}^{k+1}$. From Lemma 6.2.11 and Remark 6.2.12:

$$P\big(X_i^k = 0\big) \leq \frac{1}{2}^{X_{3i}^{k+1}+X_{3i+1}^{k+1}+X_{3i+2}^{k+1}}.$$

All the indicator variables at the same depth, $\{X_i^{k+1} | 1 \leq i \leq 3^{k+1}\}$, are independent and identically distributed. Denote the probability that $X_i^{k+1} = 1$ as p_{k+1}. Thus

$$\big(X_{3i}^{k+1} + X_{3i+1}^{k+1} + X_{3i+2}^{k+1}\big) \sim \text{Bin}(3, p_{k+1}).$$

So

$$P\big(X_i^k = 1\big) = p_k \geq 1 - \sum_{j=0}^{3} \frac{1}{2}^j \binom{3}{j} (p_{k+1})^j (1 - p_{k+1})^{(3-j)}.$$

Assume that $p_{k+1} \geq 1/2$ then:

$$\begin{aligned}
p_k \geq & 1 - \sum_{j=0}^{3} \frac{1}{2}^{j} \binom{3}{j} (p_{k+1})^j (1 - p_{k+1})^{(3-j)} \\
\geq & 1 - \sum_{j=0}^{3} \frac{1}{2}^{j+3} \binom{3}{j} \\
= & 1 - \frac{1}{8}\left(1 + \frac{3}{2} + \frac{3}{4} + \frac{1}{8}\right) \\
= & 1 - \frac{27}{64} \\
\geq & \frac{1}{2}.
\end{aligned}$$

From the definition $p_d = 1 \geq 1/2$ so from induction $p_1 \geq 1/2$ and thus with probability greater than $1/2$ there is a path from the root to a node at depth d. □

Theorem 6.2.14. *The sequence* $\text{iter}(V)$ *achieves an expected unfairness of* $\Omega(\log n)$ *for the* Randomized Global Greedy *algorithm when run on a clique.*

Proof. From Lemma 6.2.13, after the *Randomized Global Greedy* algorithm processes $\text{iter}(V)$ there exists with probability $\geq 1/2$ a path between the root and a vertex at depth $\Omega(\log n)$. Thus there exist sets such that each iteration the value of u increases by one. After $\Omega(\log n)$ iterations there exists a set such that $u = \Omega(\log n)$. □

VII Summary

7.1 Our Results

We have shown three main results in this thesis:

1. We introduced an adaptive adversary that achieves unfairness of at least $n/4$ for *any randomized algorithm*, where n is the number of leaves.

2. We presented a sequence that achieves a lower bound of $\Omega(\sqrt{n})$ for the *Star Algorithm* when the social network is a star.

3. We showed a sequence that achieves a lower bound of $\Omega(\log n)$ for *Randomized Global Greedy* when the social network is a clique.

In addition, we presented a "toolbox" of sequences which were used to prove the lower bounds for the oblivious adversaries.

7.2 Open Questions

There are several outstanding open questions regarding bounds for oblivious adversaries that follow from this thesis:

- Does *Randomized Global Greedy* achieve $\Theta(\log n)$ on the clique?

- Is the lower bound of $\Omega(\log n)$ for *Randomized Global Greedy* also correct on the star?

- Is there also an upper bound of $O(\log n)$ for *Randomized Global Greedy* on the star?

- Is there another algorithm that achieves $O(\log n)$ either on the clique or on the star?

- Does there exist an algorithm that achieves $O(1)$ on a significant class of graphs?

Bibliography

[1] M. Ajtai, J. Aspnes, M. Naor, Y. Rabani, L. J. Schulman, and O. Waarts. Fairness in scheduling. *Journal of Algorithms*, 29(2):306–357, 1998.

[2] Carma. Carma carpooling - your commuting rideshare companion. carmacarpool.com.

[3] R. Fagin and J. H. Williams. A fair carpool scheduling algorithm. *IBM Journal of Research and development*, 27(2):133–139, 1983.

[4] E. Fehr and K. M. Schmidt. A theory of fairness, competition, and cooperation. *Quarterly journal of Economics*, pages 817–868, 1999.

[5] L. Festinger. *A Theory of Cognitive Dissonance*. Mass communication series. Stanford University Press, 1962.

[6] A. Fiat, A. R. Karlin, E. Koutsoupias, C. Mathieu, and R. Zach. Carpooling in social networks.

[7] G. F. Loewenstein, L. L. Thompson, and M. H. Bazerman. Social utility and decision making in interpersonal contexts. *Journal of Personality and Social Psychology*, 57(3):426–441, 1989.

[8] M. Naor. On fairness in the carpool problem. *Journal of Algorithms*, 55(1):93–98, 2005.

[9] R. Tijdeman. The chairman assignment problem. *Discrete Mathematics*, 32(3):323 – 330, 1980.

[10] Waze. Ridewith by waze. www.waze.com/ridewith/.

תקציר

דמיינו קבוצה של אנשים שבכל יום זוג מהם רוצים לנסוע למקום משותף (למשל למקום עבודתם). בבעלות כל אחד מן האנשים רכב, אך הם מעדיפים לחסוך כסף ולנסוע ביחד. כיצד יוחלט מי ייקח את רכבו וינהג בו, ומי ייסע איתו? בעיה זו נקראת בעיית הסדר ההסעות (carpooling בלעז).

נסתכל על המקרה המקוון (online), כלומר כשלא ידוע מראש בכל יום מי זוג האנשים שירצו לנסוע. על המנגנון להחליט על סמך הבקשות הקודמות ועל סמך הבקשה הנוכחית בלבד, בלי ידע על הבקשות העתידיות. נתרכז במקרה בו רשימת הבקשות נוצרת על ידי יריב שיודע כיצד המנגנון פועל, אך לא יודע את ההגרלות האקראיות של המנגנון (אם קיימות כאלה).

כיצד נמדד מנגנון? לאחר שהמנגנון טיפל בכל הבקשות, ניתן להגדיר חוסר שיוויון עבור כל אדם כערך המוחלט של הפרש כמות הפעמים שהסיע וכמות הפעמים שנסע. על מנגנון למזער את המספר הזה עבור כל אדם בקבוצה.

נתעניין בשני תרחישים שונים, תרחיש בו אפשרי שכל זוג שחקנים יסעו יחדיו, ותרחיש בו יש אדם קבוע שנוסע כל יום ויש מספר אנשים שיכולים להצטרף אליו.

עבודה זו מציגה שלושה חסמים תחתונים:

- במקרה בו יש אדם קבוע, ועבור כל מנגנון שלא מקבל החלטות אקראיות נראה חסם תחתון של $\Omega(n)$.
- במקרה בו יש אדם קבוע, ישנו מנגנון שמשיג ביצועים קבועים במקרה בו הבקשות מתפלגות בצורה אחידה (ולא נקבעות על ידי יריב). אנו נראה חסם תחתון של $\Omega(n^{0.5})$ עבור מנגנון זה במקרה בו הבקשות מגיעות מיריב.
- עבור המנגנון החמדן כאשר כל זוג אנשים יכולים לנסוע יחדיו נראה חסם של $\Omega(\log n)$.

כל החסמים שנראה יהיו חסמים קונסטרוקטיביים, כלומר נראה סדרה של בקשות שמשיגה חסם זה.

אוניברסיטת תל-אביב

הפקולטה למדעים מדויקים ע"ש ריימונד ובברלי סאקלר

בית הספר למדעי המחשב ע"ש בלבטניק

חסמים תחתונים לבעיית הסדר ההסעות

חיבור זה הוגש כעבודת מחקר לקראת התואר "מוסמך אוניברסיטה" במדעי המחשב

על ידי

רותם צח

בהנחייתו של פרופ' עמוס פיאט

אב התשע"ה