On the Core of the Multicommodity Flow Game Without Side Payments

by

Coulter Beeson

B.Sc. University of British Columbia, 2017

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

August 2019

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

On the Core of the Multicommodity Flow Game Without Side Payments

submitted by **Coulter Beeson** in partial fulfillment of the requirements for the degree of **Master of Science** in **Computer Science**.

Examining Committee:

Bruce Shepherd
Supervisor

Hu Fu
Supervisory Committee Member

Abstract

In this thesis we investigate the core, or the set of stable solutions, of a cooperative game without side payments. This "multicommodity flow" game was initially introduced by Papadimitriou to model incentives in internet routing. He posed the questions of whether such stable solutions exist and if we can find them. Markakis and Saberi showed both how to solve this game in the setting with side payments, and that such stable solutions always exist with or without side payments. Yamada and Karasawa provide an algorithm for generating one core solution in the special case when the underlying transit graph is a path (as well as special cases of trees). We provide a new algorithm for generating more core elements and a method to test core membership efficiently in this same special case. We empirically characterize this larger set of solutions. The data suggests that stability does not necessarily impede efficiency, and that there is little trade-off between efficiency and fairness.

Lay Summary

It is often the case that a network, such as a communication network, is owned by multiple entities aiming to maximize their own profits. For instance different Internet Service Providers own the hardware that runs the internet and must share it to satisfy their individual customers demands. Our work can be summarized as answering a special case of the question "How can we share the Internet?" Currently such sharing agreements are built in an ad hoc way and lack theoretical guarantees. We can model such interactions as a cooperative game. In this thesis we provide a method for finding stable sharing agreements in particularly simple networks when players are not allowed to encourage others to take specific agreements through the use of side payments (bribes). We run simulations to investigate the diversity of these stable agreements with a focus on how productive they are for all of society and how fair they are to the individuals.

Preface

This thesis is an original, unpublished work by Coulter Beeson, written under the supervision of Bruce Shepherd.

Contents

List of Figures

Acknowledgments

I would like to thank Bruce Shepherd for his guidance, insight, and thoughtful encouragement; my family for their unwavering support; and the Natural Sciences and Engineering Research Council of Canada (NSERC) without whose funding this research would not be possible.

Chapter 1

Introduction

Problems worthy of attack prove their worth by fighting back. — Piet Hein

Cooperative game theory investigates the nature of cooperation between self-interested parties when they cannot bribe (provide side payments to) others to participate. Each participant is only incentivized to accept an agreement if they can find no better offer. If there are no groups who by their collusion could secede from a given agreement, then such an agreement is stable. While the stability of a solution says nothing of the quality of an agreement, especially in terms of the distribution of the wealth produced, it gives certainty to the future which is desirable. It is a generally accepted goal to adopt stable agreements. One may then also reason about their structure and diversity, and make prescriptions as to which are in the best interest of the public.

We use this general framework to examine routing in networks where the nodes are autonomous agents. The cooperative game we focus is used to model routing internet traffic and was introduced by Christos Papadimitriou. The self interested agents (players) in this game are anyone who has a supply of capacity, granted by their ownership of the hardware, and a desire to route some traffic for their customers. We further assume that traffic can only be routed through this network via the cooperation of these players. That is to say: if a player rejects the current agreement they simply can not route others' traffic. Thus by sharing their capacity each player can have access to a larger network and can satisfy more of their own

demand.

The rest of the section is dedicated to formalizing these ideas.

1.1 Coalition Games

A *coalition game* without side payments consists of a set of players V and a *characteristic function* Π which maps each $S \subseteq V$ to a subset of $\mathbb{R}^V_{\geq 0}$. These are sometimes referred to as games with *non-transferable utility* (NTU). The interpretation is that $\Pi(S)$ denotes the set of possible payoff (utility) vectors π available to players of S if they decide to cooperate (we assume that $\pi_i = 0$ if $i \notin S$). A general theme in cooperative game theory is to find strategies whereby the *grand coalition*, namely V itself, becomes a stable set of partners. In other words, we seek a payoff vector $\pi \in \Pi(V)$ such that no proper subset S of the players could do better if they secede from the grand coalition. We call such a coalition S a *breakaway set* for π; we now define this notion formally.

For two vectors $x, y \in \mathbb{R}^N$, we say x *dominates* y *on* S if $x_i > y_i \ \forall i \in S$; we may write $x >_S y$. Let S be a proper subset of V and $\pi \in \Pi(V)$. We call a second payoff vector $\pi' \in \Pi(S)$ a *S-deviation from* π if $\pi' >_S \pi$; we sometimes refer to the strategies inducing this payoff as a deviation. We call S a *breakaway set of* π if there is some *S-deviation*.

The *core* of a coalition game is the set of payoff vectors $\pi \in \Pi(V)$ which have no breakaway sets. We sometimes refer to the strategies inducing these core payoffs, as core strategies. Thus core vectors represent payoffs π to the grand coalition which are stable in the sense that no subset of players is motivated to defect from the strategy which induces π.

We can define the core formally then as the following set,

$$\text{core} = \{\pi \in \Pi(V) : \forall S \subset V,\ \pi' \in \Pi(S)\ \Rightarrow \exists v \in S,\ \pi_v \geq \pi'_v\}$$

It is clear from this definition that increasing the payoffs cannot remove a vector from the core (subject to feasibility).

1.2 Multiflow Coalition Game

A *multicommodity flow coalition game* is a coalition game where the players are the nodes of a *supply graph* $G = (V,E)$; we refer to $e \in E$ as a supply edge. In addition, we are given *capacities* $c(v)$ on each node $v \in V$. We are also given a *commodity graph* $H = (V,F)$. We refer to $e \in F$ as a commodity edge associated with a *demand* $d_e \geq 0$.

Strategies in a flow coalition game arise from feasible flows in G for the commodities H. For each commodity $uv \in F$, we denote by $\mathscr{P}_{uv}$ the set of (simple) paths in G joining u, v. We denote the sum of all flow between two nodes as $f_{uv} = \sum_{P \in \mathscr{P}_{uv}} f(P)$. We denote the sum of all flow terminating at a player v as $f_v = \sum_{u \in V} f_{uv}$. A *flow* is then defined by a non-negative assignment of flow f_P to each path P joining the endpoints of some commodity. Formally, f is a feasible flow if it satisfies:

1. $\sum_{P \ni v} f_P \leq c_v \quad \forall v \in V \qquad$ *Capacity constraint*
2. $f_{uv} \leq d_{uv} \quad \forall uv \in F = E(H) \qquad$ *Demand constraint*
3. $f_P \geq 0 \quad \forall P \in \mathscr{P}_{uv}\, \forall uv \in F = E(H)$

We let $\mathscr{F}(G,c,H)$ denote the set of all feasible flows. We are also interested in the strategies available to a subset S of players. We denote by $\mathscr{F}(G[S],c,H[S])$, or simply $\mathscr{F}^S$, the feasible flows in $G[S]$, the induced subgraph of G on S.

Each feasible flow induces *payoffs* (or *utilities*) for the players as follows. For any $S \subseteq V$ and $f \in \mathscr{F}^S$, we define $\pi_v(f) = f_v$ as the sum of all flows that terminate at v (we make no distinction between traffic coming from or going to v). Note that if $v \notin S$, then $\pi_v(f) = 0$. The set of payoffs available to a coalition is thus $\Pi(S) = \{\pi(f) : f \in \mathscr{F}^S\}$. The set of feasible flows is a lower comprehensive convex polytope and the utility function is linear so $\Pi(S)$ is also a lower comprehensive convex polytope. A polytope Γ is lower comprehensive if $y \leq x$ and $x \in \Gamma$, $y \in \Gamma$.

We can define for each coalition S the set of payoffs on which they can do better by seceding $D(S) = \{\pi \in \mathbb{R}^V_{\geq 0} : \exists \pi' \in \Pi(S)\ \pi' >_S \pi\}$. Then clearly $D(S) \subset \Pi(S)$, and we can rewrite the definition of the core as $\Pi(V) \setminus \bigcup_{S \subset V} D(S)$. From this

definition we can see it is possible for the core to be non-convex. Indeed Figure 1.1 illustrates just that.

We use Figure 1.1 to illustrate notation for representing flows that we re-use throughout this thesis. Straight edges represent actual edges of the supply graph, curved edges represent demand edges and are labelled x/y where x represents the amount of flow sent and y the demand for that commodity. Nodes are labelled x/y where x is the amount of capacity used and y is the initial amount of capacity available (capacities are omitted when possible for clarity). If a commodity is fully routed it is drawn in bold, if an edge is not routed it is drawn as a dashed line and when a player has no more remaining capacity we draw them with a bold circle. Each players payoff is listed below them. The flow indicated in Figure 1.1 has a symmetric version which produces a second utility vector in the core $x = (1,2,1,2),\ y = (2,1,2,1)$. The $\frac{1}{2}x + \frac{1}{2}y$ convex combination produces $(\frac{3}{2},\frac{3}{2},\frac{3}{2},\frac{3}{2})$ which is not in the core since the two central nodes would deviate by fully routing their shared commodity resulting in a pay off of $(0,2,2,0)$.

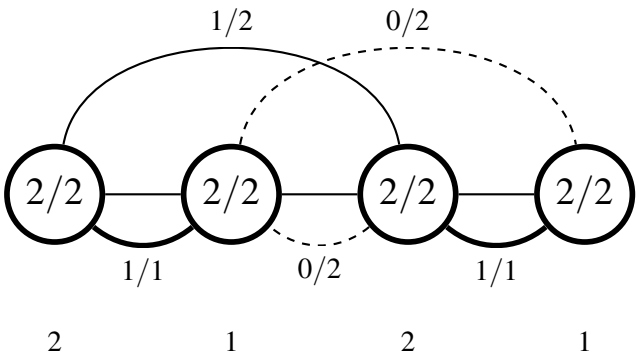

Figure 1.1: Example of non-convexity of the core

1.3 Related Work

The Multicommodity Flow Game was originally introduced by Papadimitrio to model incentives in internet routing [5]. In this model each node or player represents an Autonomous System (AS). Each AS is a component of the internet administrated by (usually) one operator, such as a university or corporation. Each

AS then desires to selfishly route as much of traffic originating from its users as possible. On the other hand ASs must cooperate with each other to allow their customers access to the board network.

While we focus on games without side payments or *non-transferable utility* (NTU) games, much of game theory has focused on games with side payments or *transferable utility* (TU) games. In TU games we assume that players can losslessly transfer utility between each other. This can be achieved by an equally valued currency. However, this is is often not the case in practice. For instance, when utility is tied to the players identity like fame or grades in school, when side payments are illegal (bribery), or when players have vastly different amounts of wealth and do not equally value the payouts. One example of NTU games are Stable Matching problems where the payout a player receives is determined by their preference for the player they are matched with. Then stable matchings are exactly those matchings whose payoff is in the core. It is well known that for bipartite graphs the core is always non-empty and a core element can be found using the Gale-Shapley algorithm. However when the graph is not bipartite the core need not exist.

Formally a TU utility game is a set of players V and a characteristic function $\pi : \mathscr{P}(V) \to \mathbb{R}_{\geq 0}$ where the interpretation is that π maps coalitions to single payoff. We can take any NTU game and convert it to a TU game in a canonical way. Let (V, Π) be a NTU game, then define $\pi(S) = \max_{\pi' \in \Pi(S)} \left\{ \sum_{i \in S} \pi'_i \right\}$. We call $\sum_{i \in V} \pi_i$ the *social welfare* (SW) of the payout π. We can analogously define the core of a TU game as the set of all payoff vectors $\pi' \in \mathbb{R}^V_{\geq 0}$ such that $\sum_{i \in V} \pi'_i = \pi(V)$ and $\sum_{i \in S} \pi'_i \geq \pi(S)$.

In TU games the strategy is fixed at the social welfare maximizing outcome and the difficulty is in determining how to split the profits. Whereas in NTU games the strategy to choose is non-obvious but the allocation of profits is completely fixed. Indeed not all core allocations achieve the same social welfare, and those that maximize social welfare often can be highly unfair.

In both cases of TU or NTU games we run into the issue that the core does not always exist. Scarf however offers a sufficient condition for the core to be non-empty. To introduce this we first define *balanced collections*. A collection of coalitions $T \subseteq \mathscr{P}(V)$ is *balanced* if there exists weights w_S such that for every player i, $\sum_{S \ni i} w_S = 1$. If all weights are in $\{0, 1\}$, then this is exactly a partition of

V. We say that a payoff π is *attainable* by S if $\pi \in \Pi(S)$. A game is *balanced* if for all balanced collections T, if π^S is attainable for all $S \in T$ then π is attainable for the grand coalition. Scarf's theorem then says that every balanced game has a non-empty core [7].

Markakis and Saberi [2] show that the TU Multicommodity Flow Game has a non-empty core and how to compute such an element in polynomial time by solving the dual program. They also show that the NTU Multicommodity Flow Game is balanced with a simple proof that we repeat here for completeness.

Lemma 1. *The multicommodity flow game with non-transferable payoff is balanced and hence has a non-empty core [2]*

Proof. Let T be a balanced collection with weights w_S and π be a payoff such that π^S is attainable for all coalitions $S \in T$.

Every coalition $S \in T$ must have a flow $f^S \in \mathscr{F}^S$ such that $\sum_{j \in S} f^S_{ij} = \pi_i \; \forall i \in S$. We can then construct a flow $f = \sum_{S \in T} w_S f^S$, such that $\sum_{S \ni i} w_S \sum_j f^S_{ij} = \sum_{S \ni i} w_S u_i = u_i$ thus u is achievable for V and the game is balanced. □

While there are numerous constructive proofs of Scarf's Lemma, Goldberg, Papadimitriou, and Savani suggest that implementing these methods in general is PSCACE-Complete [1].

For the multicommodity flow game on a path Yamada and Karasawa overcome this difficulty and provide an efficient combinatorial algorithm for producing a single core element [8]. To state their algorithm succinctly we first define a subroutine that we re-use as follows.

ROUTE

Input. A path $P_{k\ell}$

$m \leftarrow \min_{v \in P_{k\ell}} \{C_v\}$

$f_{k\ell} \leftarrow \min \{d_{k\ell}, m\}$

$C_v \leftarrow C_v - f_{k\ell} \;\; \forall v \in P_{k\ell}$

Where C_v denotes the residual capacity for node v. We say that a commodity $k\ell$ *crosses* a player v if $v \in (k,\ell)$. Intuitively we route as much flow for a commodity $k\ell$ as its minimum bottleneck. Where the minimum bottleneck is the lesser the smallest residual capacities of any players crossed by $k\ell$ and the demand $d_{k\ell}$.

We are now ready to present the Yamada and Karasawa algorithm.

Y&K PATH ALGORITHM

Input. Given a Game (G,c,H), where G is a path

Output. a flow f such that $\pi(f) \in Core(G,c,H)$

$f \leftarrow 0, \;\; C_v \leftarrow c_v$

for $\ell \in [2...n]$ **do**

  for $k \in [\ell...1]$ **do**

    route$\big([k,\ell]\big)$

  end

end

Return f

This algorithm essentially processes the players in a left to right fashion, by symmetry of the path we can (usually) obtain a second core element by running the algorithm from right to left. We later develop a more general algorithm to find more core elements.

1.4 Notation and Definitions

$G = (V, E)$	Graph with vertex set V and edge set E
$V(G)$, $E(G)$	Vertex and Edge set of G
$G[S]$	Induced subgraph of G for the set S
$\mathbb{R}^V$	Real vector space indexed by elements of V
$\mathbb{R}^V_{\geq 0}$	Non-negative orthant of $\mathbb{R}^V$
$\mathscr{P}(V)$	The power set of V
$x(S) = \sum_{v \in S} x_v$	The sum of the elements of x in S
V	The grand coalition
$S \subset V$	S is a subset of V, S is a coalition
$\mathscr{F}^S$	Set of feasible flows for coalition S
$\mathscr{G} = (G, c, H)$	A multicommodity flow game
c_v	Capacity of player v
C_v	The residual capacity of player v
d_{uv}	Demand between players u and v
f_P	the flow along path P
f_{uv}	Sum of all flow on paths between players u and v
f_v	Marginal flow of v
$\mathscr{P}_{uv}$	Set of all paths connecting u and v
P_{uv}	Unique path between u and v when the graph is a tree
$[k, v]$	Unique path between u and v when the graph is a path
$I(P)$	The interior of the path P, i.e. the path without its end-points
$\pi(f)$	Payoff vector for flow f
π^S	Payoff π projected onto the coalition S
$\Pi(S)$	Set of all payoffs for coalition S

Chapter 2

Certificates

In this section we consider the question of how to show that a given coalition cannot secede from the grand coalition. All results in this section are for general graphs. Let $\hat{f}$ be some feasible flow. How can we check if the payoff vector $\pi(\hat{f})$ is in the core? Let us consider a more restricted question. For a given set S, can we verify that it has no deviation? That is, can we certify that there is no flow $f \in \mathscr{F}^S$ such that $\pi(f) >_S \pi(\hat{f})$? The existence of such a deviation corresponds to feasibility of the following strict-inequality linear program.

$$\begin{array}{lll} \text{maximize} & 0^T f & \\ \text{subject to} & \sum_{P \ni v} f_P \leq c_v & v \in S \\ & \sum_{P \in \mathscr{P}_{k\ell}} f_P \leq d_{k\ell} & k\ell \in H[S] \\ & \pi_v(f) > \pi_v(\hat{f}) & v \in S \\ & f \geq 0 & \end{array} \tag{2.1}$$

To show that S does not have a deviation f, we must show that the above linear program is infeasible. By Farkas' Lemma, a linear system $Ax \leq b, x \geq 0$ is infeasible if and only if $\exists y \geq 0$ such that $y^T A \geq 0$ and $y^T b < 0$ (for further details and proofs of Farkas Lemma see [6]). This would immediately give a way to show that a coalition S is not a breakaway, but for the fact that (2.1) has strict inequalities. Hence, for some $\varepsilon > 0$, consider a standard LP, call it $LP(\varepsilon)$, obtained by modifying the *payoff constraints* to be: $-\pi_v(f) \leq -\pi_v(\hat{f}) - \varepsilon$. We may now

apply Farkas' Lemma to this new program to see that it is infeasible if and only if there is a solution $(y_v : v \in S), (z_{k\ell} : k\ell \in E(H)), (w_v : v \in S)$ to the following system.

$$\begin{aligned} \sum_{v \in S} y_v c_v + \sum_{k\ell \in H[S]} z_{k\ell} d_{k\ell} &\leq \sum_{v \in S} w_v (\pi_v(\hat{f}) + \varepsilon) && (2.2) \\ \sum_{v \in V(P)} y_v + z_{k\ell} &\geq w_k + w_\ell \qquad \forall P \in \mathscr{P}_{k\ell},\ k\ell \in H[S] && (2.3) \\ y, z, w &\geq 0 && (2.4) \end{aligned}$$

where we use that if $P \in P_{k\ell}$, then f_P only contributes to the payoffs π_k and π_ℓ.

These facts now are used to prove the following.

Lemma 2. *S does not have a deviation if and only if there exists $y, z, w \geq 0$ which satisfy (2.3) and*

$$\sum_{v \in S} y_v c_v + \sum_{k\ell \in E(H[S])} z_{k\ell} d_{k\ell} \leq \sum_{v \in S} w_v \pi_v(\hat{f}) \tag{2.5}$$

and such that $w \neq 0$.

Proof. Note that S does not have a deviation if and only if $LP(\varepsilon)$ is infeasible for any $\varepsilon > 0$. First suppose that y, z, w satisfy the prescribed conditions. Since $w_v > 0$ for some v, we then have that the strict inequality holds (2.2), for any choice of $\varepsilon > 0$. That is, $LP(\varepsilon)$ is infeasible for any $\varepsilon > 0$. Conversely, if S does not have a deviation, then $LP(1)$ is infeasible and hence there is an associated dual certificate y, z, w. Since the left-hand side of (2.2) is non-negative, and $\pi(\hat{f}) \geq 0$. we must have that $w_v > 0$ for some $x \in S$. □

We call a feasible solution y, z, w to (2.5), (2.3), and (2.4) a (Farkas) *certificate* for a coalition S and a payoff $\pi(\hat{f})$. Such a solution certifies both that $\pi(\hat{f})$ is in the core, and that S cannot deviate from the strategy $\hat{f}$. We call such a coalition *certifiable* for the payoff $\pi(\hat{f})$. We construct our certificates from the flow $\hat{f}$ so we sometimes also say S is certifiable under $\hat{f}$. To establish that some payoff vector is in the core, we must argue that any coalition S is certifiable. To establish correctness of the algorithm we develop we must show that every coalition is certifiable

for any flow returned.

We note some additional properties of certificates.

Without loss of generality, we restrict our attention to coalitions S such that $G[S]$ is connected. This is without loss of generality as if S has a deviation, then each individual connected component of $G[S]$ must also have a deviation.

Lemma 3. *If there are polynomially many connected subgraphs, then we can efficiently certify that a payoff is in the core.*

Proof. Lemma 2 implies there are only polynomial many linear programs to solve each of which can be solved in polynomial time. □

Lemma 4. *If a coalition S is certifiable for a given payoff $\pi(\hat{f})$, then S is certifiable for any payoff arising from a flow $f \in \mathscr{F}^V$ such that $f \geq \hat{f}$.*

Proof. Clearly payoffs are monotonic with regard to flow, and the payoffs only occur in constraint (2.5). As increasing the payoff can only improve this constraint, S is still certifiable under f. □

2.1 Implied Values

Given vectors $y, w \geq 0$ we can easily determine if there is z for which y, z, w is a certificate, since the "best" choice of z is for each $k\ell \in H[S]$, to set

$$z_{k\ell} := Z_{k\ell}(y,w) = \max_{P \in \mathscr{P}_{k\ell}} \left\{ w_k + w_\ell - \sum_{v \in V(P)} y_v, 0 \right\}. \tag{2.6}$$

We refer to $Z_{k\ell}(y,w)$ as the value *implied* by y, w. This is the best value since $z_{k\ell}$ only occurs in 2 constraints: in (2.3) where it needs to be at least the implied value, and (2.5) where making it smaller improves the inequality.

Lemma 5. *If y, z, w is a certificate, then so is y, z', w where z' is set to the values implied by y, w.*

From now on, we assume that z is set to the implied values. Hence we only need to check condition (2.5).

Our certificates have a special structure in that the vectors y, w are binary. We call a node *tight* if it has no remaining residual capacity. Then the support of y is a set Y of tight nodes, which is a subset of the support of w. The first few results handle the straightforward cases where certificates are trivial to construct.

We call a node v *globally content* if $\pi_v = c_v$. We hardly need a certificate to show that no feasible flow (for any coalition S!) could yield strictly greater utility for v. Nevertheless we construct a certificate as a warm-up for later results.

Lemma 6. *If there exists a globally content player $v \in S$, then S is certifiable.*

Proof. We construct the certificate as $\{v\} = Y = W$. Note that the right hand side of (2.3) is only non-zero for demands which terminate at v. For such demands $y_v = 1$ already handles this constraint. Then the implied value for $z = 0$.

Now for condition (2.5) we have

$$\begin{aligned} yc + zd &= c_v \\ &= \pi_v \\ &= w\pi \end{aligned}$$

□

We now assume there are no globally content players and hence:

Assumption 1. *Every tight player in S transits some flow.*

If there exists $v \in S$ such that $\pi_v \geq \sum_{u \in S} d_{uv}$, then no feasible flow from $\mathscr{F}_S$ can route more flow to v. We say that such a player v is *S-content*.

Lemma 7. *If there exists an S-*content *player, then S is certifiable*

Proof. We set $w_v = 1$ and $y = 0$. Hence the implied values are $z_{kv} = 1$ for all commodities $kv \in H[S]$.

Again we check condition (2.5).

$$\begin{aligned} yc + zd &= zd \\ &= \sum_{kv \in H[S]} d_{kv} \\ &= \sum_{kv \in H[S]} f_{kv} \\ &\leq \sum_{k \neq v} f_{kv} \\ &= \pi_v \\ &= w\pi \end{aligned}$$

Where the second equality follows from z only selecting commodities incident v, and the third equality follows as every commodities (from $H[S]$) incident v is fully routed (i.e. $d_{kv} = f_{kv}$)

□

Henceforth we assume the following.

Assumption 2. *Every player in S must be incident to a commodity within $H[S]$ that is not fully-routed.*

Note that if there are no tight nodes in S, then every player in S is S-content. Hence we assume the following.

Assumption 3. *The coalition S contains a tight node.*

2.2 Pre-Certificates and Anchor Sets

Our certificates are based on two sets of nodes $Y \subseteq W$, where Y is a set of tight nodes. We call this pair a *pre-certificate* if the vectors $y = \mathbb{1}_Y, w = \mathbb{1}_W$ can be extended to a certificate by setting a vector z to the implied values (2.6). We can

think of the variables y, z, w as selecting some node capacities, demands, and node utilities respectively. We then think of the selected demands and capacities as costs that need to be charged against the selected utilities. i.e. to verify the condition (2.2) we ensure $w\pi - (yc + zd) \geq 0$. In other words, a *charging scheme* consists of paying for the selected capacities yc and demands zd by using the payoffs for the selected players $w\pi$.

Next we determine sufficient conditions for sets Y, W to be a pre-certificate. But first we introduce the notion of anchors.

We define the *anchors* of a node v as $A(v) = \{k \in V : \exists P \in \mathscr{P}_{k\ell},\ f(P) > 0,\ v \in I(P)\}$. That is, if there is positive flow routed over a player v, then the endpoints of the flow are anchors of v. Given a coalition S we refer to any positive commodity with exactly one end point in S as an *external commodity*.

Lemma 8. *Consider a pair of node sets $Y \subseteq W$ where nodes in Y are tight. This forms a pre-certificate if the following properties hold:*

- ***P1)*** *every commodity with both endpoints in W is fully routed.*
- ***P2)*** *every positive commodity that crosses a player $v \in Y$ has an endpoint in W*
- ***P3)*** *every non-fully routed commodity with an endpoint in W crosses or terminates at a node in Y*
- ***P4)*** *no positive flow crosses or touches more than one node in Y*

Proof. Observe that by our construction of the certificate $y_v, w_v \in \{0, 1\}$. For each $k\ell \in H[S]$, we define $W(k, \ell) := w_k + w_\ell$ and so by (2.6) $z_{k\ell} = \max_{P \in \mathscr{P}_{k\ell}} \Big\{ w_k + w_\ell - \sum_{v \in V(P)} y_v, 0 \Big\}$. It follows that $z_{k\ell} \in \{0, 1, 2\}$ and by **P1** & **P3**, that $z_{k\ell} > 0$ only if kl is a fully-routed demand.

It now remains to verify (2.5). We do this by a charging argument. First, we charge demands selected by z to utilities of chosen players in $W \setminus Y$.

Case 1) $z_{k\ell} = 2$:

Then $W(k,\ell) = 2$ and $k\ell$ does not cross or touch any node of Y. By **P1**, $k\ell$ is fully-routed and we charge this "twice selected" demand against both endpoints.

Case 2) $z_{k\ell} = 1$:

There are two subcases, either $W(k,\ell) = 1$ or $W(k,\ell) = 2$.

Case 2.1) $z_{k\ell} = 1$ and $W(k,\ell) = 1$:

Then $k\ell$ neither touches or crosses a node in Y. **P3** thus ensures that any such commodity is fully-routed, and we charge the $z_{k\ell}d_{k\ell}$ to the endpoint in W.

Case 2.2) $z_{k\ell} = 1$ and $W(k,\ell) = 2$:

By definition of $z_{k\ell}$, the commodity $k\ell$ must cross or touch precisely one node in Y. In the case that $k\ell$ terminates at a node in Y, then we charge the demand $z_{k\ell}d_{k\ell}$ to the utility of its endpoint *not* in Y. Otherwise $k\ell$ crosses a node in Y and **P1** ensures $k\ell$ is fully routed. In this case we arbitrarily charge one endpoint k or ℓ.

Notice that the demands considered in Case 2.2) contribute utility to two nodes in W and we have charged its $z_{k\ell}d_{k\ell}$ to one of them. Similarly if $z_{k\ell} = 0$ for a fully-routed demand crossing a player in Y, then $W(k,\ell) = 1$ (if $W(k,\ell) = 2$, then property **P4** ensures that $z_{k\ell}$ is implied to be 1) and we have not yet charged the utility from the chosen endpoint. We denote the total such surplus s.

We have now accounted for all demands selected by z, i.e., we have charged the quantity zd. We must now account for yc using the remaining utility from $w\pi$. Let's classify the unspent utilities. First, we have only charged players in $W \setminus Y$, Second, since $z_{k\ell} > 0$ only for fully-routed demands, we have not yet charged any utility from split demands. We have not used utility from external demands. Finally, as noted earlier we sometimes have surplus utility s associated with one of the endpoints of a fully-routed demand (see discussion after Case 2.2). For each

type of surplus we denote its total excess utility as follows: s (as defined above), σ (from split commodities), e (from external commodities) and $\sum_{v\in Y}\pi_v$ (from players in Y). We thus have that $w\pi - zd \geq s+\sigma+e+\sum_{v\in Y}\pi_v$.

We must now charge the quantity yc to the surplus utilities. Since any positive commodity touches or crosses at most one node of Y (by **P4**) we consider each node in Y separately and then charge the demands which used its capacity. For a given node $v \in Y$, the quantity $y_v c_v$ arises in two ways: as flow that terminates at v or as flow that crosses v. We charge all of the capacity of c_v used by flows terminating at v to π_v. The remaining capacity is used by positive commodities crossing v. By **P2** we have that every crossing commodity has at least one endpoint in W. First, if this is a split commodity we charge its unique endpoint in W; note that this flow contributed to σ. So it remains to consider a crossing commodity $k\ell$ which is fully-routed. Either $z_{k\ell} = 0$ or $W(k,\ell) = 2$ (commodities from Case 2.1 do not cross any players in Y). In both cases it contributed to s, and so there is an available endpoint in W which has not been charged (by this commodity).
Combining these observations we deduce that

$$yc \leq s+\sigma+\sum_{v\in Y}\pi_v.$$

Combining this with our earlier observation that $w\pi - zd \geq s+\sigma+e+\sum_{v\in Y}\pi_v$ we have

$$w\pi - zd - yc \geq e \geq 0$$

and hence the implied certificate y,z,w satisfies (2.2).

□

Armed with the above lemmata and assumptions we are now ready to move on to defining our algorithms.

Chapter 3

Algorithms

We consider the following natural class of algorithms which greedily routes flow for commodities according to a given partial order $\preceq$ on the pairs of players. We focus on the special case where there is a unique path connecting any two players k, ℓ. That is to say, when the supply graph is a tree. In this case we denote the path between to players k, ℓ as $P_{k\ell}$ or in when the supply graph itself is a path $[k, \ell]$. Given that paths are unique we simplify our notation and write $f_{k\ell} = f_{P_{k\ell}}$. After a pair of players is considered in the algorithm we call that pair *visited*. Let $\mathscr{M}(f)$ be the set of minimal un-visited pairs for the current flow. Where we mean minimal with respect to our partial order $\prec$.

POSET-GREEDY
Input. Given a Game (G, c, H, d), and an order $\preceq$
Output. a flow f such that $\pi(f) \in Core(G, c, H, d)$
$f \leftarrow 0, \;\; C_v \leftarrow c_v$
while $\exists\, k\ell \in \mathscr{M}(f)$ **do**
  if $k\ell \in E(H)$ **then**
    route$\big(P_{k\ell}\big)$
  end
end
Return f

Notice that the Y&K PATH algorithm can be viewed as a POSET-GREEDY algo-

rithm with a total ordering of pairs of players. One property of the ordering used in the Y&K PATH algorithm is that if $[k\ell] \subset [ij]$ then $k\ell \prec ij$. More generally we call an ordering *nested* if for any given path P it routes all subpaths of P before P itself. We first note some properties flows returned by the POSET GREEDY algorithm.

For the following let f be any flow returned by the POSET GREEDY algorithm

Lemma 9. *If $f_{k\ell} < d_{k\ell}$, then $\exists v \in V(P_{k\ell})$ that is tight*

Proof. By definition of the greedy algorithm such a v must exist otherwise $f_{k\ell} = d_{k\ell}$. □

Lemma 10. *For a given slack node v let the $K(v)$ be the maximal connected subgraph containing v that contains no tight nodes. Then every commodity in $E(H[K])$ is fully routed.*

Proof. Suppose there exists such a commodity, then Lemma 9 requires there be a tight node in $K(v)$, a contradiction. □

3.1 Properties of Nested Flows

In the following let f be an arbitrary flow returned by POSET-GREEDY using a nested ordering.

Lemma 11. *If $f_{k\ell} > 0$, then $f_{ij} = d_{ij}$, for all $P_{ij} \subset P_{k\ell}$*

Proof. If $f_{k\ell} > 0$, then prior to this commodity being added $C_v > 0 \quad \forall v \in P_{k\ell}$. Because all commodities ij such that $P_{ij} \subset P_{k\ell}$ are processed prior to $k\ell$ due to the nested order after ij was routed $\nexists v \in P_{ij}$ such that $C_v = 0$. That is there was no tight node when ij was routed otherwise $f_{k\ell} = 0$. □

Lemma 12. *If $f_{k\ell} < d_{k\ell}$, then $f_{ij} = 0$, for all $P_{ij} \supset P_{k\ell}$*

Proof. If $f_{k\ell} < d_{k\ell}$, then there is a tight node $x \in P_{k\ell}$. Thus for all commodities ij the node x blocks any flow as they are routed later in the nested ordering. □

We partition the commodities into three sets $F^0, F^<, F^=$ *zero commodities*, *split commodities*, and *fully routed commodities*. Formally,

$$F^0 = \{k\ell : 0 = f_{k\ell} < d_{k\ell}\}$$
$$F^< = \{k\ell : 0 < f_{k\ell} < d_{k\ell}\}$$
$$F^= = \{k\ell : 0 < f_{k\ell} = d_{k\ell}\}$$

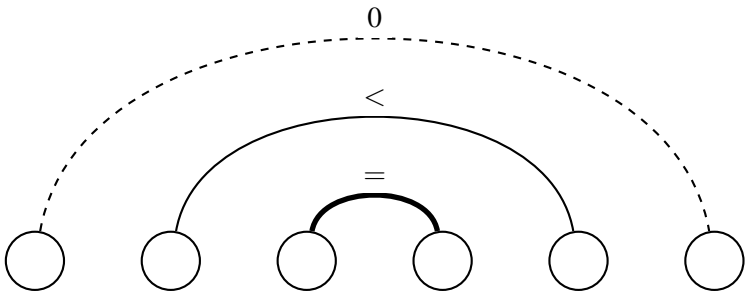

Figure 3.1: Nested Property

We summarize the main properties of nested flows to aid intuition:

- For $k\ell \in F^< \cap F^=$: $P_{ij} \subseteq P_{k\ell} \Rightarrow ij$ is fully-routed
- For $k\ell \in F^<$: $P_{ij} \supseteq P_{k\ell} \Rightarrow ij$ is a zero commodity
- No split commodity is a subset of another split commodity.
- Call a commodity ij *maximal* if $ij \notin F^0$ and for any commodity $k\ell$ such that $P_{k\ell} \supset P_{ij}$, $k\ell \in F^0$. Then every split commodity is maximal.
- Let ij be a maximal commodity. Then (i) $P_{k\ell} \subset P_{ij} \Rightarrow k\ell$ is fully-routed, and (ii) every commodity $k\ell$ such that $P_{k\ell} \supset P_{ij}$ is a zero commodity.

Lemma 13. POSET GREEDY *with a nested ordering does not always produce an element of the core.*

Proof. To illustrate this we provide a counterexample. Figure 3.2 shows a flow resulting from a nested ordering. The payoff of each player is listed below it. We

can then easily see in second part of the figure that the middle four players have a breakaway.

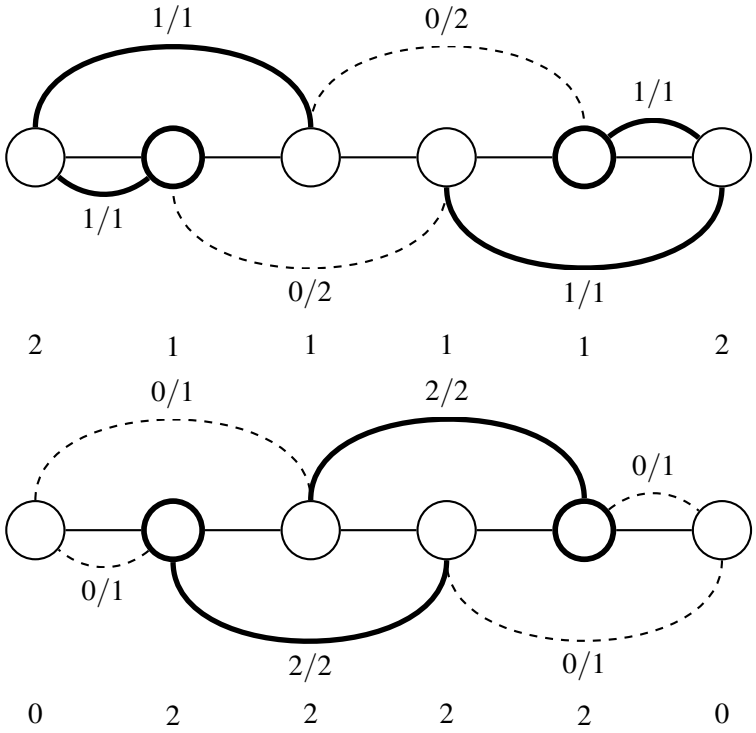

Figure 3.2: Counterexample to General Nested Ordering

□

3.2 Incorporation Algorithm

We now present our main algorithm we call an *Incorporation Algorithm*

INCORPORATION ALGORITHM
Input. Given a Game (G,c,H,d), and a nested order $\preceq$
Output. a flow f such that $\pi(f) \in Core(G,c,H,d)$
$f \leftarrow 0, \;\; C_v \leftarrow c_v, \;\; S \leftarrow \{r\}$
while $S \subset V$ **do**
  Let $v \in N(S)$
  $S \leftarrow S + v$
  while $\exists\, k\ell \in \mathscr{M}(f) \cap H[S]$ **do**
    if $k\ell \in E(H[S])$ **then**
      route$\left(P_{k\ell}\right)$
    end
  end
end
Return f

Informally the algorithm grows an *incorporated* set of players by adding neighbouring players one at a time. When a player is incorporated we route all commodities incident that player with its other end point also in the incorporated set according to our given partial order $\prec$. The terminology is especially fitting as both "incorporate" and "core" etymologically derive from the Latin *corpus*. At any point in the execution of the algorithm all players in this set are fully incorporated, then the currently flow is in the core of this sub-game. That is to say, every subset of the incorporated set is certifiable.

Assuming n is odd, then it is not hard to verify that there are $2\sum_{i=1}^{\frac{n-1}{2}} \binom{n-(i+2)}{i-1}$ different orderings in which we could incorporate players each potentially giving rise to different core element. Thus we can hope to generate exponentially many different core elements from our algorithm.

Chapter 4

Paths

In this section we focus on the case when the transit graph itself is a path. We then consider a flow f returned by the INCORPORATION ALGORITHM using a nested ordering. In order to prove correctness we must show that an arbitrary coalition $S = [i, j]$ is certifiable. Formally,

Lemma 14. *If $\hat{f}$ is a flow returned by the* INCORPORATION ALGORITHM *with a nested ordering, then $\pi(\hat{f})$ is in the core.*

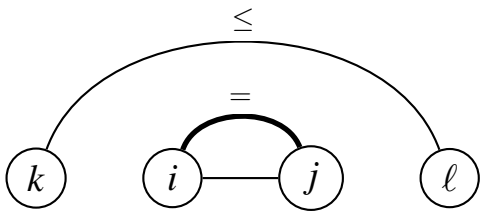

Figure 4.1: Configuration of Lemma 15.

Lemma 15. *If $G[S]$ is a path and there is a positive commodity $k\ell$ such that $S = [i, j] \subseteq [k, \ell]$, then S is certifiable.*

Proof. If such a commodity exists and $k\ell \neq ij$, then by Lemma 11, every commodity in $H[S]$ is fully-routed. But then every player in S is S-content, contradicting our assumption. If $k\ell = ij$, then every node in (i, j) is S-content as well. This only leaves the case $j = i+1$. We must have $i(i+1) \in F^{<}$ or else both i and $i+1$

would be S-content. But then since f was produced via a nested order, $i(i+1)$ was routed (and hence blocked), before any demands which transit through either i or $i+1$. It follows that one of i or $i+1$ hit capacity and is hence globally content, contradicting our assumption. □

We now further refine our notion of anchors for paths. For a given node v we split its anchors into disjoints sets of *left* and *right anchors* $L(v) = \{u \in V : u \in A(v),\ u < v\}$ and $R(v) = \{u \in V : u \in A(v),\ v < u\}$ respectively. We define the left (resp. right) *anchor sets* for a player v as $Y = \{v\}$, and $W = L(v) + v$. (resp. $Y = \{v\}$, and $W = R(v) + v$).

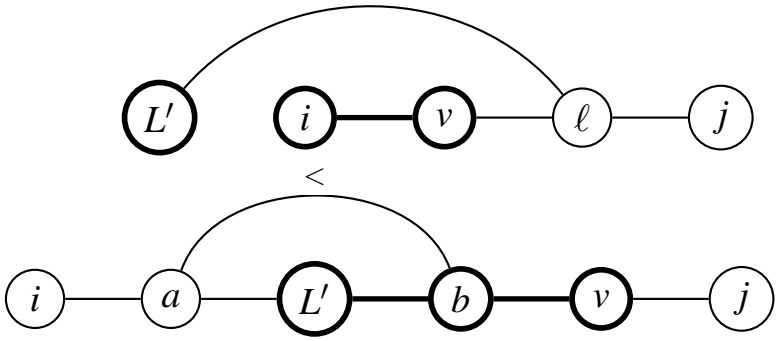

Figure 4.2: The configuration disallowed by **PA** and **PB** (resp.)

Lemma 16. *The left (resp. right) anchor sets of a tight node $v \in S$ is a pre-certificate as long as the following conditions hold:*

PA) *$L(v) \subset S$ (resp. $R(v) \subset S$)*

PB) *there is no non-fully routed commodity ab such that $b \in L(v)$ and $a \in [i, L')$ (resp. $a \in R(v)$ and $b \in (R', j]$).*

Where L' is the leftmost anchor of v (resp. R' is the rightmost anchor of v).

Proof. We show that properties **PA** and **PB** ensure that all properties from Lemma 8 are satisfied.

P1) First, note that W is a subset of $[L', v]$ where L' is the leftmost anchor. Any commodity with both endpoints in W is thus nested under a positive commodity that terminates at L' and crosses v. Hence by Lemma 11 it is fully routed.

P2) By **PA** any positive commodity that crosses v must have its left endpoint in $[i, v)$. Hence by definition, its left endpoint is a left anchor which is then in W.

P3) Suppose that ab is a non-fully routed commodity which does not cross or terminate at v. By Lemma 11, we must have $a < L'$ or else it is fully-routed. But then this possibility is disallowed by **PB**.

P4) Follows trivially as there is only one player in Y.

Lemma 8 now implies that Y, W is a pre-certificate.

□

Lemma 17. *If v is a leftmost tight node in S, then all commodities ab such that $[a, b] \subseteq [i, v)$ are fully routed (resp. rightmost)*

Proof. The proof follows from Lemma 10. □

Now consider the execution of the INCORPORATION ALGORITHM after the pair i, j is visited we have the flow $\hat{f}$. We use Lemma 4 to argue that the coalition S is certifiable for the final flow returned by the INCORPORATION ALGORITHM.

Lemma 18. *If $\hat{f}$ is the flow resulting after i, j is visited, then $[i, j]$ is certifiable under the flow $\hat{f}$.*

Proof. Without loss of generality, let i be the player currently being incorporated. By Assumption 3 there must be at least one tight node, w.l.o.g. let v be a leftmost tight node.

We now show that the left anchor sets of v form a pre-certificate by Lemma 8

PA) The player i is only just being incorporated so no commodity with an endpoint farther left than i has been routed so $L(v) \subset [i, j]$.

PB) As v is a leftmost tight node we can apply Lemma 17.

Thus $[i, j]$ is indeed certifiable. □

We now complete the proof of Lemma 14.

Proof. When the algorithm terminates the last pair it considers are the endpoints of the path, and all subsets have been considered prior. By Lemma 18 when a pair is consider is corresponding subset $[k, \ell]$ was certifiable for the current flow. By Lemma 4 after potentially adding more flow subsequently each set remains certifiable. Thus every coalition of contiguous players is certifiable for $\hat{f}$, and $\pi(\hat{f})$ must be the core. □

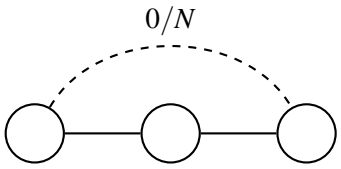

Figure 4.3: Example Non-maximal Core Flow

Thus we have demonstrated an algorithm for finding numerous core elements for games on the path. Furthermore the flow returned by the INCORPORATION ALGORITHM is a maximal flow. However as we can see in Figure 4.3 a flow need not necessarily be maximal to be in the core. In this instance the zero flow is in the core, whereas routing the only commodity can result in a maximal flow with arbitrarily higher social welfare. In Figure 4.4 we see two core elements with different social welfare. The first flow has a social welfare of 16 while the second has a social welfare of 14. However, the second flow has a minimal payoff of 1, while the first has minimal payoff of zero. This is the so called efficiency-equity trade-off.

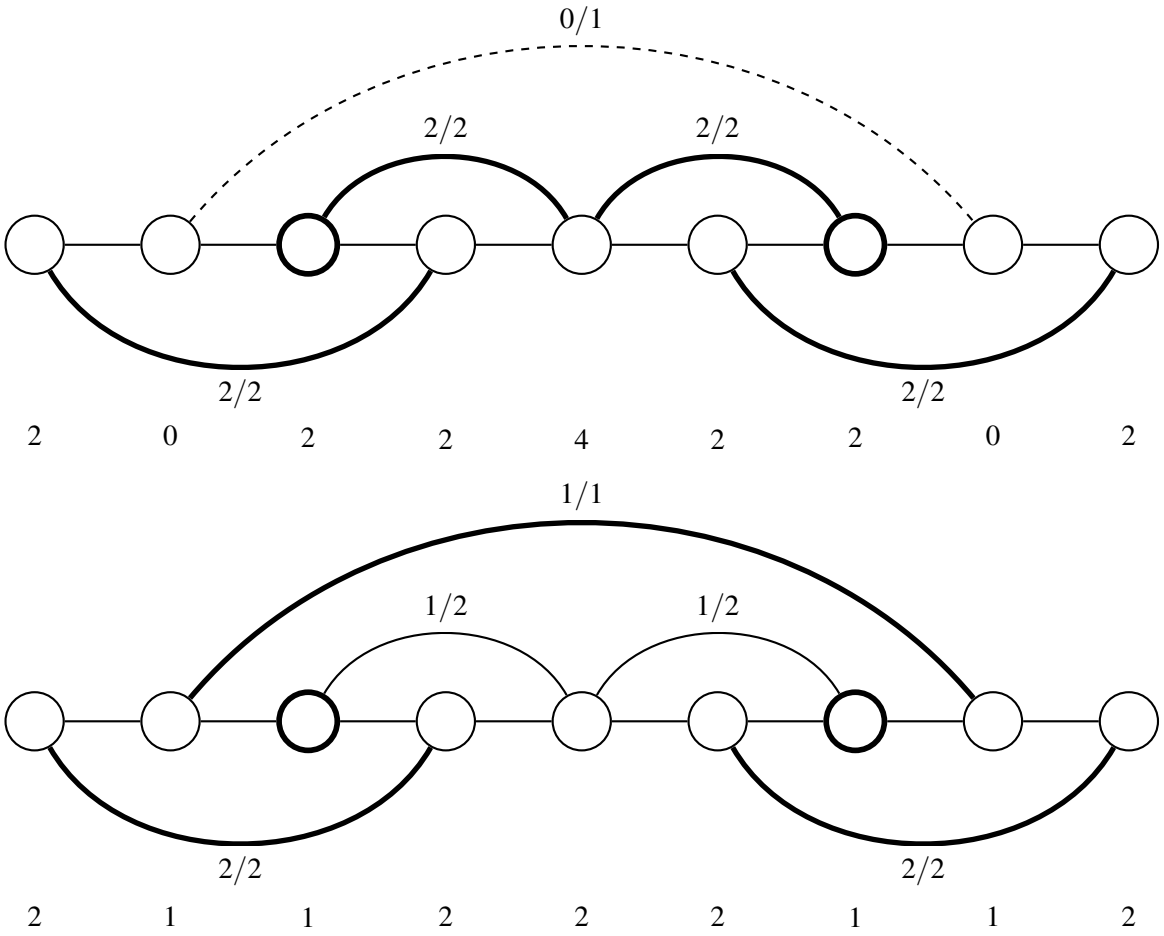

Figure 4.4: Example Efficiency Equity Trade-off

4.1 Rooted Paths and Spiders

Consider running the INCORPORATION ALGORITHM with a nested ordering on a path, but ignoring all commodities that cross r. We claim that this also produces an element of the core.

It is easy to see that coalitions which do not contain r are unaffected so lets consider a coalition $[i, j]$ such that $r \in (i, j)$. But even in this case the arguments from Lemmamata 18 and 14 still apply as the certificates are unaffected by these

un-routed commodities. Thus the resulting flow is still in the core although not necessarily maximal. Essentially, for every coalition either r is S-content or we can use a leftmost (rightmost) tight node as a certificate (see proof of Lemma 14).

This idea can easily be extend to spiders. A *spider graph* is a tree with at most one node with degree greater than two. That is it is a star with each ray replaced by an arbitrary path. We call this distinguished node in the center the root of the spider, and each maximal path with the root as an endpoint is a *leg* of the spider. Yamada and Karsawa provide an algorithm for finding core solutions for spiders [8]. Consider running the INCORPORATION ALGORITHM with a nested ordering where we set the root of the spider as the starting node r. Then the above argument still holds and we can safely ignore any commodities that cross r. Again, for every coalition either r is S-content, or at least one of the legs of the spider has a deepest node that can be used to create an anchor certificate. We now make this idea formal.

Lemma 19. *The* INCORPORATION ALGORITHM *always returns a core flow, when the transit graph is a spider and r is the root of the spider.*

Proof. Any coalition $[i, j]$ not containing r is routed the same as if we ran the Y&K PATH ALGORITHM w.l.o.g on the subset $[r, j]$ alone. We no consider a coalition S that contains r. Consider the flow that results after the last player was incorporated into S. By Assumption 3 there is a tight player in S. If r is tight, then r is globally content and we are done. So assume the tight player is on one of the legs. W.l.o.g. let v be a maximally deep tight node (i.e. a tight node such that there is no tight node farther away from r on the same leg), and let t be the deepest node on the same leg as v. Then we can consider the path P_{rt} in isolation and again Lemma 18 directly applies. □

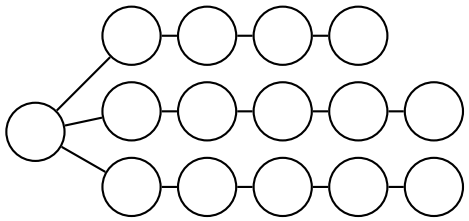

Figure 4.5: Example Spider Graph With 3 Legs

In either case all the remaining commodities may be routed in any arbitrary way, so as to improve the fairness and/or social welfare of the allocation.

Chapter 5

Trees

In this section we attempt to extend our analysis of the path and spiders to general trees. However, we have thus far been unable to produce an efficient algorithm for finding core elements. Nonetheless we present here some counterexamples to natural orderings, and partial results as we have found them.

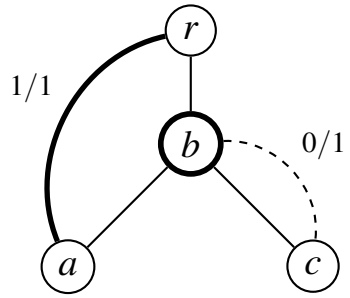

Figure 5.1: Counterexample to Nested Incorporation Order on a Tree

First, we consider simply applying the INCORPORATION ALGORITHM with a nested ordering. Figure 5.1 however shows a counterexample. If we grow our incorporated set from r to b, then to c we route the commodity ra using up all of the capacity of b. However the coalition $\{b,c\}$ has a deviation as they could simply route their single commodity. It seems using b's capacity to transit flow before handling a coalition such as $\{b,c\}$ is problematic.

One natural set of orderings is to use a tree decomposition. That is, we choose a root and then recursively route subtrees and combine them. However, we can

easily adapt our counterexample from Figure 3.2 as illustrated in Figure 5.2. This ordering does provide a core element to the instance in Figure 5.1.

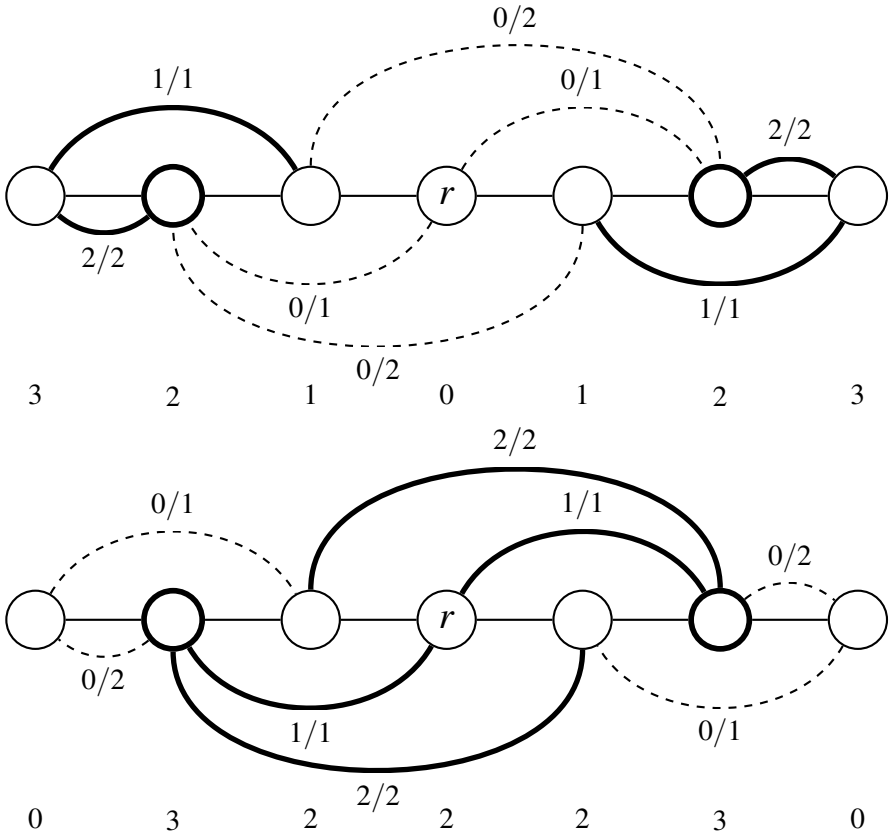

Figure 5.2: Counterexample to Recursive Tree Routing

Observe that in Figure 5.2 if there were no commodities that crossed r, then routing subtrees recursively would indeed be successful. This leads us to a simple lemma but first a definition. Assume that we have an arbitrary out r-arborescence for the tree T. As before let P_{ij} denote the unique path from i to j in T, and D_{ij} be the path with edges directed as in the arborescence. We then classify commodities as either *vertical* if D_{ij} is a directed path, or *horizontal* otherwise.

Lemma 20. *If there exists only horizontal commodities, then the zero flow is in the core.*

Proof. Consider a potential breakaway S. Let r_S be the root of the subset S. Then r_S is S-content. □

We now extend our notion of left and right anchor sets from the path to the tree.

Again, we partition the anchors of a player $A(v)$ into two sets the up and down anchors of v denotes $U(v)$ and $D(v)$ respectively. Where $U(v)$ is the set of all anchors that are ancestors of v relative to the root r and $D(v)$ is the set of anchors that are descendants of v.

Lemma 21. *If v is a maximally deep tight node in S , then there does not exist any non-fully routed commodity ab such that such that a and b are both deeper than v.*

Proof. The claim follows from Lemma 10. □

Lemma 22. *The down anchor sets of a tight node $v \in S$ are a pre-certificate as long as the following conditions hold:*

PA) $D(v) \subset S$

PB) *Every commodity ab such that a and b are both deeper than v, and at least one of a or b is in $D(v)$ is fully routed.*

Proof. We show that the given properties ensure all properties of Lemma 16 are satisfied.

P1) First, note that every node in W is a descendent of v. By assumption there are no horizontal commodities so consider a vertical commodity ab with both endpoints in W w.l.o.g. let b be deeper than a. By virtue of b being an anchor of v there must exist some positive commodity of routed along P_{vb}, but $P_{ab} \subset P_{vb}$ so ab is nested under a positive commodity and must be fully routed.

P2) By **PA** any positive commodity that crosses v has a deepest endpoint in $D(v) \subset W$.

P3) Suppose ab is a non-fully routed commodity with an endpoint in W that does not cross of terminate at v. Then both ab are deeper than v and by **PB** must be fully routed.

P4) There is only player in Y.

□

Lemma 23. *If S contains a deepest tight node v and all its down anchors $D(v)$, then S is certifiable.*

Proof. By assumption **PA** of Lemma 22 is satisfied as $D(v) \subset S$.

v is a deepest tight node and so by Lemma 21 all commodities with endpoints deeper than v are fully routed and thus property **PB** is satisfied as well. □

Lemma 24. *Let P_{rv} be a maximal path in S and $k\ell$ a positive commodity such that $P_{rv} \subset P_{k\ell}$. If v's subtrees (contained in S) contain no tight nodes, then S is certifiable.*

Proof. If all of v's subtrees are empty, then v is S-content as all commodities incident v nest under $k\ell$.

Suppose v's subtrees are not empty, then by our extended nested ordering any commodity uv such that u is in a subtree of v must precede $k\ell$, but v was not tight when $k\ell$ was routed thus v must be fully routed on all of its subtrees and all of its ancestors. Thus v is S-content. □

It is tempting to create an analogous proof for up anchor sets however here we run into a problem. Consider a coalition S and a shallowest tight node t and that the analogous properties to **PA** and **PB** hold. That is to say suppose that t's up anchors are all in S (i.e. $U(t) \subset S$). And suppose that every commodity ab such that a and b are both *shallower* than t, and at least one of a or b is in $U(t)$ is fully routed. Then an anchor can be incident a non-fully routed commodity with its other endpoint deeper than t in a subtree that does not contain t (violating property **P3** of Lemma 16). To address this we would need to include multiple tight nodes into our certificate. But here we hit a catch. Consider the following example in Figure 5.3. While in this case there are certificates of another form, if we attempt to create an

up anchor certificate we find that we must include all of t and v's anchors as well as t and v themselves in our set W. But now the commodity vr has both endpoints W, but is not fully routed (violating **P1** of Lemma 16).

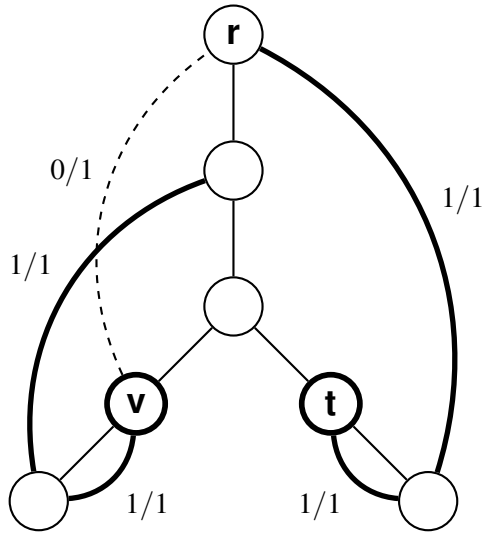

Figure 5.3: Counterexample to Up Anchor Set Certificate

We propose on final ordering along with its counterexample before ending our discussion on trees. Consider rooting the tree at a node r and decomposing the graph into level sets from r. That is to say $L_0 = \{r\}$, $L_1 = \{v : \text{depth}(v) = 1\}$, etc... We then again consider growing an incorporated set from the root, but by adding entire levels at a time. To maintain the nested ordering, when routing level k we first route all commodities between level k and $k-1$ and then $k-2$ and so on until we reach the root. Commodities between two given levels are routed in any order. Figure 5.4 shows a possible flow resulting from such an ordering and its breakaway.

It is our hope that these examples can help guide further research on finding core solutions in general trees.

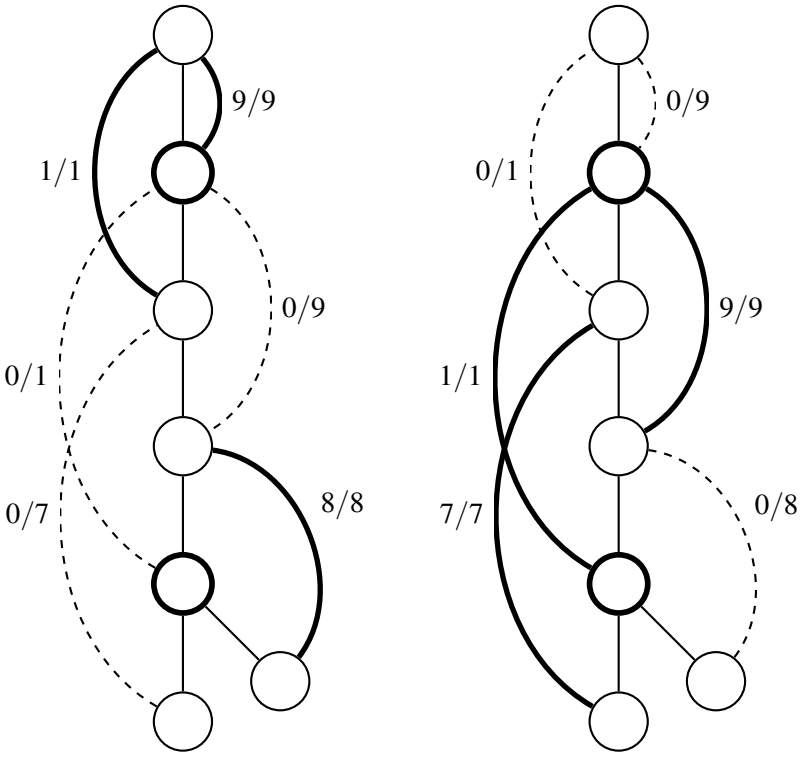

Figure 5.4: Counterexample to Level Order Routing

Chapter 6

Empirical

To those who, seeing the vice and misery that spring from the unequal distributions of wealth and privilege, feel the possibility of a higher social state and would strive for its attainment. — Henry George, Progress and Poverty (1879)

We now have an algorithm that can generate a potentially large and diverse set of stable solutions to our game. Although our algorithm only generates a subset of possible core solutions, by randomizing the order in which it executes we can sample from this subset. We call this generated subset of solutions the *Empirical Core*, denoted *EC* throughout this chapter. It is natural to study the variation amongst these solutions and to consider the question of which constitutes the best?

The problem of aggregating individual utilities (payoffs) into a social preference has long been studied and has significant implications in terms of public policy decisions and social justice. To rank different outcomes one often uses a Social Welfare objective function. That is a function that takes in a payoff vector and outputs a positive real number, where we interpret larger values as being more preferred by society. For a thorough treatment see *Fair Division and Collective Welfare* [3]. Generally we desire social welfare functions that satisfy the following conditions:

- *Monotonicity*: If a players' utility increases *ceteris paribus*, then the latter profile is preferred.

- *Symmetry*: The function is blind to the identity of players. Formally it is invariant under permutations.
- *Continuity*: For a given payoff vector π the sets of all vectors weakly better and weakly worse than π are closed (where a payoff vector x is weakly better (resp. worse) than a payoff vector y if $x_i \geq y_i$ (resp. $x_i \leq y_i$))
- *Independence of Unconcerned Agents*: The function does not depend on agents whose utility is unchanged (e.g. if our objective prefers $(1,3,10)$ to $(1,2,10)$, then it must also prefer $(1,3,1)$ to $(1,2,1)$).
- *Independence of Common Scale*: The function is invariant to scaling utilities by the same value (e.g. it wouldn't matter if we measured utility in cents, dollars, or millions of dollars).

It can be shown that any function satisfying the above properties (up to multiplicative constants) is of one of the following three form:

$$W(\pi) = \begin{cases} \sum_i \pi_i^p & p > 0 \\ \sum_i \log(\pi_i) & p = 0 \\ -\sum_i \pi_i^p & p < 0 \end{cases}$$

If we require the additional property known as the *Pigou-Dalton Principle* then $p \leq 1$. Informally the Pigou-Dalton Principle states that if the sum of utilities is held constant and there is a transfer from a better off player, to a worse off player such that the difference of their utilities decreases, then the social welfare function increases (doesn't decrease). That is our social welfare function should prefer less inequality.

Three important social welfare functions are $p = 1$, $p = 0$, and $p = -\infty$. These are known as the *utilitarian* (or Benthamite) function, the *Nash Utility* function, and the *egalitarian* (or leximin/Rawlsian) function. While the egalitarian choice as a function maps all payoff vectors to zero, it still allows us to order payoffs by instead taking p arbitrarily small. We can evaluate this ordering by sorting the individual payoffs into ascending order and prefer payoff vectors that are lexicographically smaller. We approximate this objective by the minimum payoff. For

our analysis we focus on the utilitarian objective, and the minimum payoff.

The utilitarian objective is blind to the condition of the individuals and seeks only to maximize the production of our society as a whole. From now on we refer to the value of this function as the *social welfare* (SW) or the *efficiency* of an outcome. The minimum payoff objective aims to improve the stock of the worst among us leading to increased fairness of the distribution. From now on we refer to the value of this function as the *fairness* of that outcome. As shown in Figure 4.4, it is often the case that the utilitarian and min payoff objectives are at odds with each other. On one hand, increasing the product of the whole can in some games increase the payoffs of those at the top while reducing those at the bottom of the wealth distribution. That is to say increasing the production can increase inequality. Whereas increasing the position of our most marginalized individuals can damage the production of the whole. For a more dramatic example consider the following game with only two players and two outcomes $[0.99, 100]$ and $[1, 1]$. the first outcome maximizes the SW whereas the second maximizes fairness. If we were to allow the transfer of utility, then clearly, the first outcome is to be preferred if the second player gives up even a tiny fraction of their payoff. This trade-off is known as the efficiency-equity trade-off [4].

First, we explore the question "Do core elements in general have an efficiency-fairness trade-off?" To explore this we extract two subsets of the empirical core $Min(EC) = \text{argmin}_{v \in EC}\{SW(v)\}$ and $Max(EC) = \text{argmax}_{v \in EC}\{SW(v)\}$. That is, the elements that attain the minimum and maximum SW respectively.

Secondly, we aim to understand how the set of core solutions changes as we perturb the underlying game. That is to say, how does the character of stable solutions change as we vary capacity or demand?

To answer these questions we first require a game to subject to analysis. In the following section we describe the models by which we generate specific games.

6.1 Game Models

We first recall that a path game is defined entirely by the demand graph, its weights, and the capacities of the players. To select a game we either deterministically choose these values, or we create a model from which to probabilistically sample

them. To avoid modelling the dynamics of network evolution we fix the number of players to be constant.

We choose models not for their ability to accurately reflect real world routing, as most real world routing does not take place on paths. Instead these models have been chosen for their simplicity and insights into the structure of our problem.

For each model we seek control over the relative amount of capacity and demand. By setting their relative abundance we can adjust the competitiveness of the games sampled. That is to say, if the level of aggregate demand is fixed, then when capacity is scarce there is more competition for it. If we instead look at a situation in which every player has capacity in excess, then there is no competition and the only reasonable solution is to route all demands. It is clear to see that if capacities are everywhere zero, then the only feasible solution arises from the zero flow. Thus, at either extreme end of competitiveness our solution set is reduced to a single point. It is between these two bounds that we investigate the richness of our solutions. We intentionally leave the definition of competitiveness vague as it varies depending on the specific model employed.

6.1.1 Constant Model

In this model we deterministically set all demands and capacities to some constant values C, D. That is $d_{uv} = D \; \forall u, v \in V$, and $c_v = C \; \forall c \in V$.

6.1.2 Normal Model

In the normal model we aim to draw the marginal demand $d_v = \sum_{u \neq v} d_{uv}$ for each player v according to a normal distribution described below. Each player's capacity is then a scaling of their marginal demand. To generate a game we select parameters C, D, where C is the amount by which we scale the marginal demand and D is the total demand of the system.

We take a truncated normal distribution $N(\mu, \sigma)$ over $[1, n]$, with mean $\mu = n/2$ and standard deviation $\sigma = 2\sqrt{n}$ (for a formal definition of a truncated normal distribution see Appendix A.1). We then draw D pairs of players u, v independently from N and for each pair u, v we add a unit demand. In this way acquire integral demands d_{uv}. Note that the marginal demand d_v for each player v approximates the

truncated normally distribution. In this model we aim to capture the idea that more central players are likely more "popular" (i.e. have a greater demand for traffic), and that a player's capacity depends only on their own demand (and not on other players' desire to use their capacity).

6.1.3 Bottlenecks

In addition to the above models we also investigate games where an artificial bottleneck to capacity is added in the middle of the path. Concretely, when we add a bottleneck we halve the existing capacity of the central third of players. For example, in a game with $n = 60$ players, players $[20,40]$ would have half of their previous capacity.

6.1.4 Random Graph Model

In this model we first generate a demand graph according to an Erdős-Rényi random graph model with parameter p. That is, each edge is present independently with probability p. We again choose parameters C,D where capacities for each player are chosen uniformly at random from $[1,C]$, and for each demand edge present its weight is drawn uniformly at random from $[1,D]$.

6.2 Results

For each of our models we fix the parameters which determine the relative amount of demand and number of players. We then generate a list of games of decreasing competitiveness (increasing capacity). For each of these games, we then run our algorithm with a random valid ordering to generate an EC. We then plot the relevant statistics.

6.2.1 Constant Model Results

We use the constant model to illustrate the experiments performed, and use it as a reference for explaining trends across models. For this section we fix the number players $n = 50$, and unit demands $D = 1$.

First, we investigate the number of distinct core elements our algorithm pro-

duces in Figure 6.1. The number of distinct core elements rapidly spikes as capacity increases from zero to approximately 10, before trending back down to zero as capacity increases further. At the peak we sampled 85 distinct core elements. The general trend is for the number of distinct elements to decrease, but it does so in a nearly step-wise fashion, with deep inter-step valleys. In other words, there are extended periods where the number of distinct elements is constant with increasing capacity. Then the number of solutions sharply decreases briefly before plateauing again at a level lower than the previous plateau. The length of the steps increases as capacity increases. Although not every step is perfectly flat, this is likely due to statistical noise.

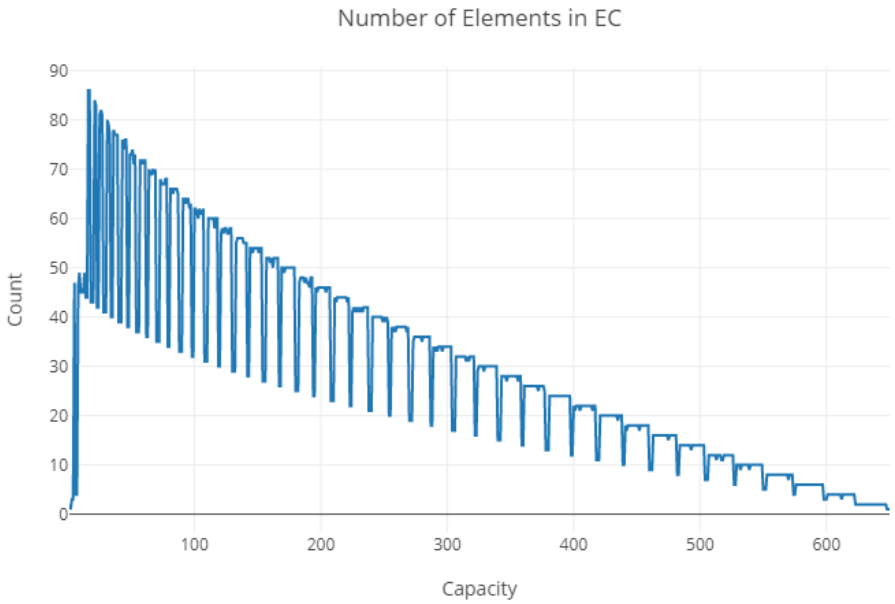

Figure 6.1: Number of Elements in EC for Constant Model, 500 samples per game

Next, for each game we plot the average of the core elements generated in Figure 6.2. It should be noted, as the core is in general non-convex, these average elements need not be in the core.

The plot shown in Figure 6.2 can be separated into two section. Those corresponding to games with a capacity above or below 200. In games with a capacity

greater than 200, on average, the more central the player the better their payoff on average. Whereas in games with a capacity below 200, we see that this is not the case. Instead as capacity decreases from 200 the positions of the richest players (symmetrically) moves towards to the endpoints. This forms ridges in the shape of a "triangle" whose base is at the endpoints of the line at zero capacity and meet at the center of the path at capacity 200.

At capacity 650 we can see that the network is saturated as every player has enough capacity to meet all of their demand (i.e. $\pi_i = \sum_{u \neq v} d_{uv} = n - 1$).

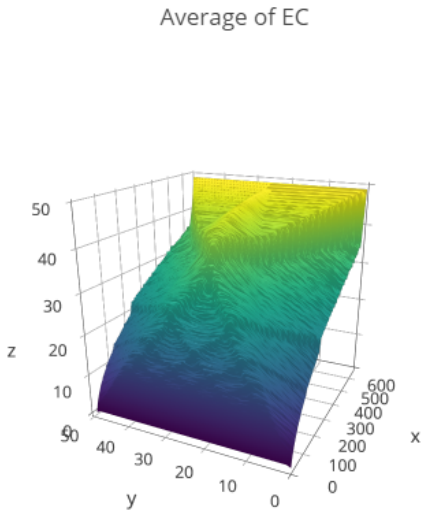

Figure 6.2: Average of EC for Constant Model, x: capacity, y: player ID, z: payoff, samples per game: 500

Next, we plot the average element of the $Min(EC)$ and $Max(EC)$ respectively in Figure 6.3. In both these plots we again see a transition at capacity 200. Unlike Figure 6.2, in the plot for the $Min(EC)$ above a capacity of 200, the central players are not richest on average. In the plot for the $Max(EC)$, the central players are richer than in the average of the EC. However, this comes at the expense of the players farther towards the endpoints.

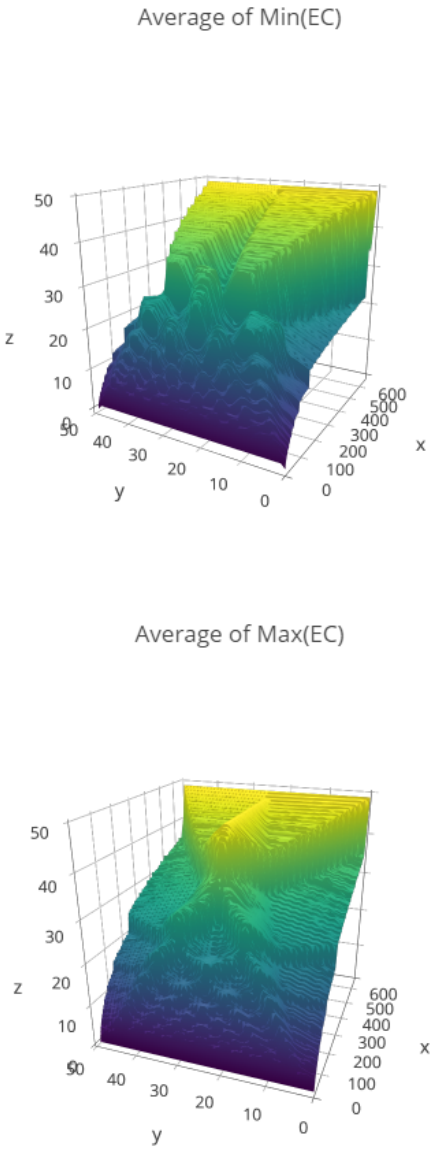

Figure 6.3: Average of Min(EC)/Max(EC) for Constant Model, x: capacity, y: player ID, z: payoff, samples per game: 500

We now compare the average SW and fairness between these three sets (The

EC, $Min(EC)$, and $Max(EC)$). In the top plot of Figure 6.4 we see SW for every set increases monotonically with capacity. In the range of $[100, 400]$ units of capacity there is a gap between the maximum SW and the minimum SW. Outside this range both essentially coincide and all core elements achieve the same SW. We can also see the above mentioned transition at capacity 200. This point corresponds to the largest gap in SW. In the second, plot fairness also increases monotonically with capacity for all three sets (modulo statistical noise from sampling). The curves for the $Max(EC)$ and $Min(EC)$ are nearly piece-wise linear as is typical of tropical curves. For all games the average fairness of the $Max(EC)$ dominates that of the $Min(EC)$. At a capacity of roughly 600 the fairness of the SW minimizing elements steeply jumps at which point all elements achieve the same fairness. In both the plots for SW and fairness the average is closer to the maximum for nearly the entire range of the plot (with the exception of a capacity of roughly 140 where the fairness of the SW minimizing elements actually greater than the average).

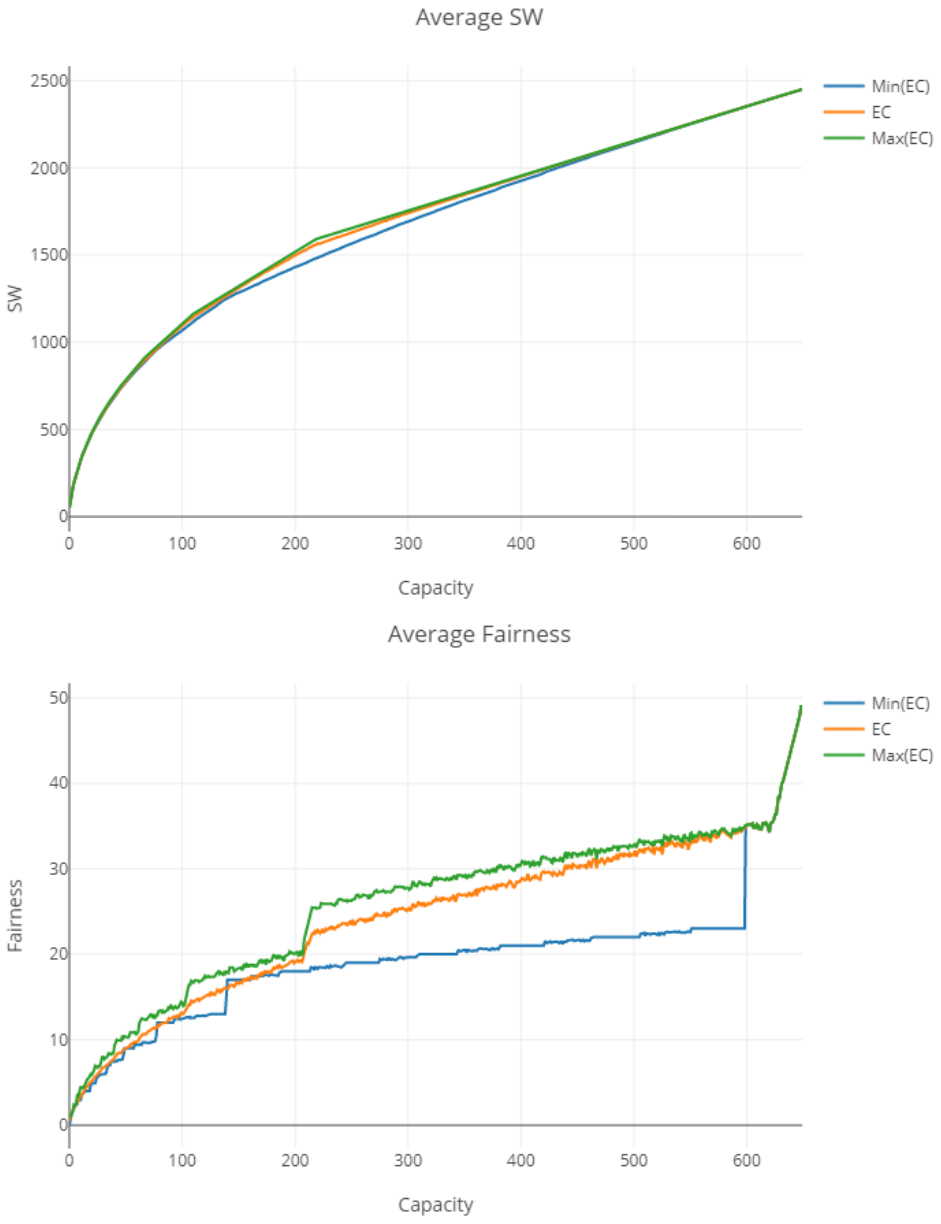

Figure 6.4: Average SW and Fairness for Constant Model, samples per game: 500

We repeat the above analysis with the addition of a bottleneck. Below the

transition point players that are in the bottleneck receive zero payoff. Besides this difference the trends in the results are similar and are included in Appendix A.2.

6.2.2 Normal Model Results

In this model we can continuously scale the amount of capacity so to sample a finite number of games we must take discrete steps. In the following plots the "capacity" refers to the scaling and not an absolute value of capacity. This means the capacity is only zero if the scaling factor is zero. Instead we choose some small arbitrary starting capacity.

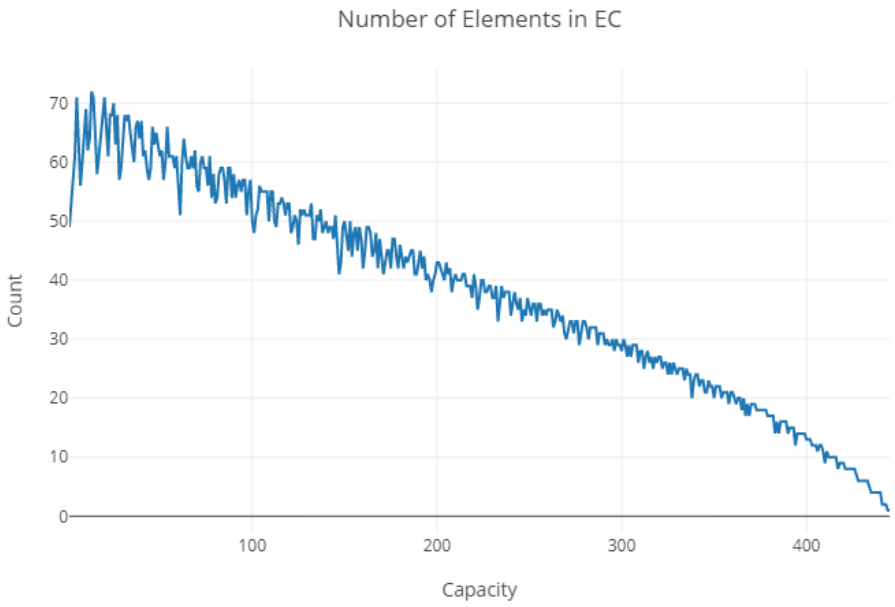

Figure 6.5: Number of Elements in EC for Normal Model, 2000 samples per game

In Figure 6.5 the general trend is again for the number of distinct elements to decrease with decreasing competition. However in this case it is unclear if there is still a periodic/step-wise character as in Figure 6.1 or if it is due to statistical noise.

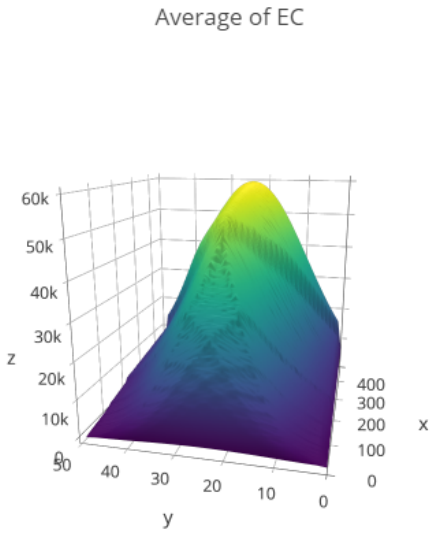

Figure 6.6: Average of EC for Normal Model, x: capacity, y: player ID, z: payoff, samples per game: 2000

In Figures 6.6 there again appears to be a transition between scarce and abundant capacity at a capacity of roughy 35. The effect of considering only the $Min(EC)$ and $Max(EC)$ is similar to that of the constant model. These plots are included in Appendix A.2.

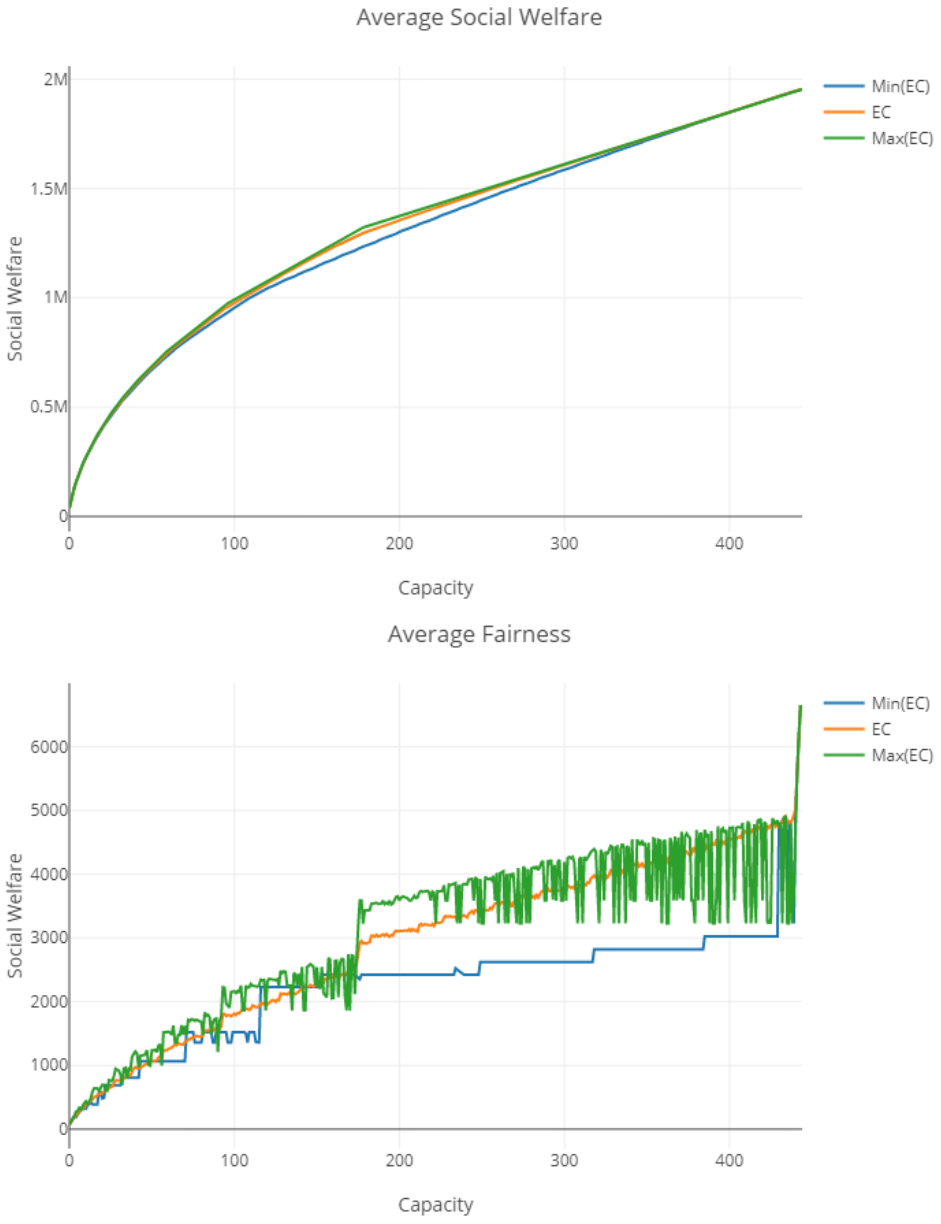

Figure 6.7: Average SW and Fairness for Normal Model, samples per game: 2000

The plots in Figure 6.7 are qualitatively similar to those for the constant model

in Figure 6.4. However, the plot for fairness is much nosier. In this model it is not the case that the average fairness of the $Min(EC)$ elements dominates the average fairness of the $Max(EC)$ elements.

Results for the inclusion of a bottleneck are again included in Appendix A.2.

6.2.3 Individual Games Constant Model

In this section we further analyze three individual games from the constant model. One where capacity is scare $C = 100$, one where it is abundant $C = 400$ and another at the transition point $C = 200$.

We again plot the average of the elements the $Min(EC)$, the $Max(EC)$, and the EC. These plots only reinforce the above observations about this transition and are included in Appendix A.2.

For the remaining plots changing the capacity did not drastically impact the results. Only those for the low capacity game are shown as and the rest are located in the Appendix A.2.

Figure 6.8 shows histograms for SW and fairness. In agreement with the previous results we see that most elements indeed maximize SW. With fairness there is a far smaller range, and while most elements don't maximize fairness the majority of elements fall on the upper half of this distribution.

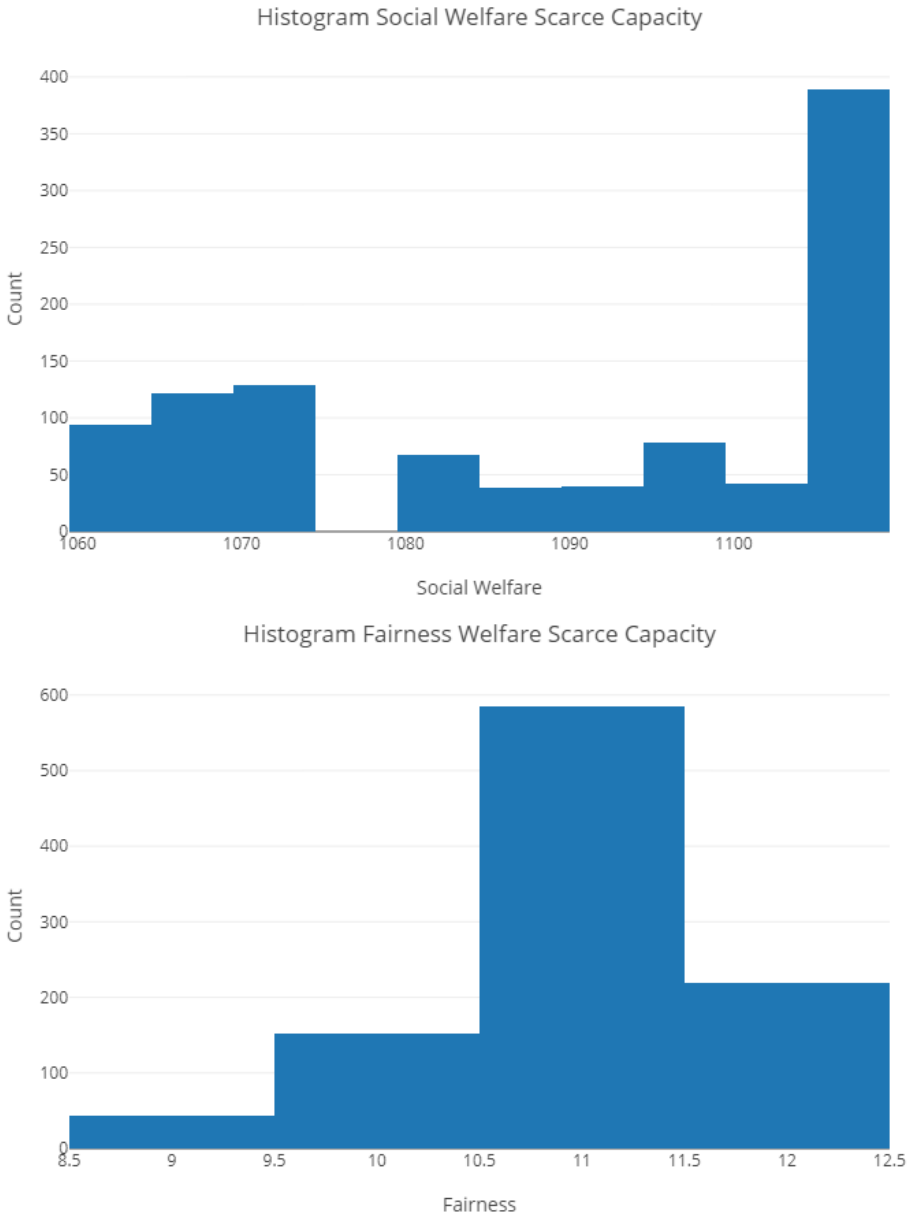

Figure 6.8: Histogram of SW and Fairness for Scarce Capacity Constant Model, Samples: 1000

Finally, we investigate the impact of the time a player is incorporated on their

payoff. In other words, if a player is Incorporated earlier do they receive more or less payoff? Surprisingly, we find that players tend to do best if they are incorporated at a time dependent on their position in the path. This effect is shown in Figure 6.9. In this figure we can see that a player k on average receives its greatest payoff if they are added nearly exactly at time k or time $n-k$, making more on average if added at the later of these two times.

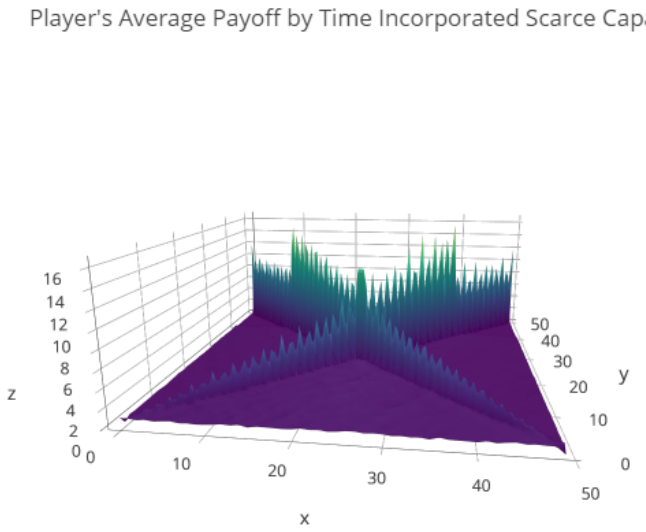

Figure 6.9: Player's Average Payoff by Time Incorporated for Scarce Capacity Constant Model, Samples: 5000, x: player, y: time incorporated, z: payoff

6.3 Discussion

From the above data we can see similar trends across models as capacity increases. In terms of the number of distinct solutions found, generally increasing capacity decreases the number of distinct solutions found. This trend is reversed when capacity is extremely scarce in which case increasing capacity rapidly increases the number of solutions found. This is not seen in the normal model due to how the data was generated. In all cases we generate substantially fewer distinct element

than the theoretical maximum. In addition to the affect on the number of elements found, increasing capacity increases both the social welfare and the fairness of the games.

Comparing across models we see that with increasing capacity, there is a transition that occurs between games with scarce capacity and games with abundant capacity. This is clearest in the graphs of average SW and fairness. At this transition point is the largest gap between the maximum SW and the minimum SW. However, even at this transition the gap is quite small. For instance, in the constant model with no bottleneck there is a maximum difference of 100 units or roughly 6% of the maximum SW. By comparison the gap between the average fairness of the $Max(EC)$ and $Min(EC)$ can be quite large. In the constant model this gap can be as large as roughly a third of the average fairness of the $Max(EC)$. Despite a larger gap, in all models except the normal model, the average fairness of the $Max(EC)$ dominates the fairness of all solutions. In all graphs the average values are much closer to the maximum than the minimum suggesting that most elements are near optimal. This conclusion is supported by the histograms produced for the individual games.

In addition to comparing the maximum SW of the EC solutions we also compared against the maximum possible over all feasible flows (i.e. the maximum multicommodity flow ignoring stability). We found that in every game for every model there exists at least one core element that matches the maximum possible SW. Which is to say that, there is no loss in efficiency from stability. To lend support to this hypothesis we also sampled 1000 games ($n = 100$) from the random graph model ($p = 0.5, C = 1000, D = 1000$) and compared the maximum possible SW to the maximum SW of all EC elements. All games compared had a core element that matched the maximum feasible SW. In addition we measured the gap between the maximum and minimum SW of EC elements and found again that in all cases the gap is small. However with these random games virtually every solutions has a player with a zero payoff and fairness cannot be improved at all.

Interestingly, it seems that there is a strong correlation between the time a player is added and their position in the network. Indeed this suggests that simply following a left to right ordering as done by Yamada and Karasawa should provide an element that approximately maximizes the payoff for every player and

thus also maximizes SW. The Yamada and Karasawa algorithm does produce an element that maximizes SW in the case of the constant model and the normal model. However, when tested on random games we find that it fails to maximize the SW. Despite this, random games still show a correlation between the time players are added and their payoff.

In conclusion the data suggests that if the objective is to maximize SW, then even the worst solution in the *EC* nearly approximates the maximum possible social welfare. Hence a random solution is likely to be near optimal. In most cases maximizing SW also maximizes fairness showing a lack of efficiency fairness trade-off. Future work should investigate whether there always exists a core solution that achieves the maximum possible SW or whether there are instance where any core solution does strictly worse.

Chapter 7

Conclusions

We have partially succeeded in our goal of producing efficient algorithms for generating core solutions for the multicommodity flow game and examining their diversity. We have developed methods for generating a large number of solutions when the underlying graph is a path or spider. In general we can certify any payoff for a multicommodity flow game, but not necessarily efficiently. However, any specific coalition can be certified or provided a breakaway in polynomial time. In the case of general trees we fall short of a general algorithm to produce core solutions, but identify structures that any poset greedy approach must deal with if it is to be successful. The case of finding core elements for general graphs we leave open.

Even in the case of paths however we fail to fully characterize the structure of core solutions. There are core elements that our algorithms can never produce (such as by violating the nested ordering), but there are also core elements that no post greedy algorithm can produce. Specifically our algorithm only returns maximal flows, however not every core element needs to be maximal. However from the perspective of maximizing SW this is a non-issue, buy it is a theoretically interesting question and could lead to techniques for optimizing over core solutions.

Empirically we confirm that we do indeed generate a diverse set of core solutions. From the perspective of the two social objective considered most solutions are nearly optimal. Additionally, there does not appear to need to be a trade-off between stability and efficiency, and the trade-off between efficiency and fairness also

seems to be small or non-existent. These claims are however empirical in nature and the data we generate are fundamentally limited by the computational power available for this thesis. In terms of optimizing solutions from the perspective of individual players more work needs to be done.

We have provided a method of generating solutions and guidance on selecting the best among them for paths (and spiders). We hope to apply these techniques to generate stable solutions for trees and general graphs. While our algorithm is centralize and uses perfect information the internet is decentralized and players are not necessarily truthful. To address this in the future we aim to develop decentralized mechanisms for finding core solutions.

Bibliography

[1] P. W. Goldberg, C. H. Papadimitriou, and R. Savani. The complexity of the homotopy method, equilibrium selection, and lemke-howson solutions. *ACM Trans. Econ. Comput.*, 1(2):9:1–9:25, May 2013. ISSN 2167-8375. doi:10.1145/2465769.2465774. URL http://doi.acm.org/10.1145/2465769.2465774. → page 6

[2] E. Markakis and A. Saberi. On the core of the multicommodity flow game. *Decision Support Systems*, 39(1):3–10, mar 2005. doi:10.1016/j.dss.2004.08.003. URL https://doi.org/10.1016/j.dss.2004.08.003. → page 6

[3] H. Moulin. *Fair division and collective welfare*. MIT press, 2004. → page 35

[4] A. M. Okun. *Equality and efficiency: The big tradeoff*. Brookings Institution Press, 2015. → page 37

[5] C. Papadimitriou. Algorithms, games, and the internet. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing STOC 01*. ACM Press, 2001. doi:10.1145/380752.380883. URL https://doi.org/10.1145/380752.380883. → page 4

[6] K. Roos. *Farkas lemmaFarkas Lemma*, pages 995–998. Springer US, Boston, MA, 2009. ISBN 978-0-387-74759-0. doi:10.1007/978-0-387-74759-0_175. URL https://doi.org/10.1007/978-0-387-74759-0_175. → page 9

[7] H. E. Scarf. The core of an n person game. *Econometrica: Journal of the Econometric Society*, pages 50–69, 1967. → page 6

[8] T. Yamada and K. Karasawa. On finding a solution in the core of a multicommodity flow game on a spider. In *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, pages 1011–1014. IEEE, 2006. → pages 6, 27

Appendix A

Supporting Materials

A.1 Truncated Normal Distribution Definition

Let $N(\mu, \sigma)$ be a normal distribution. Let $\phi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right)$, and $\Phi(x) = \frac{1}{2}\left(1 + \text{erf}(x/\sqrt{2})\right)$ be the probability density function and cumulative distribution function for the standard normal distribution respectively. Then a truncated normal distribution with a lower bound of L and an upper bound of U has the following probability density function:

$$f(x) = \frac{\phi\left(\frac{x-\mu}{\sigma}\right)}{\sigma\left(\Phi\left(\frac{U-\mu}{\sigma}\right) - \Phi\left(\frac{L-\mu}{\sigma}\right)\right)}$$

A.2 Supporting Results

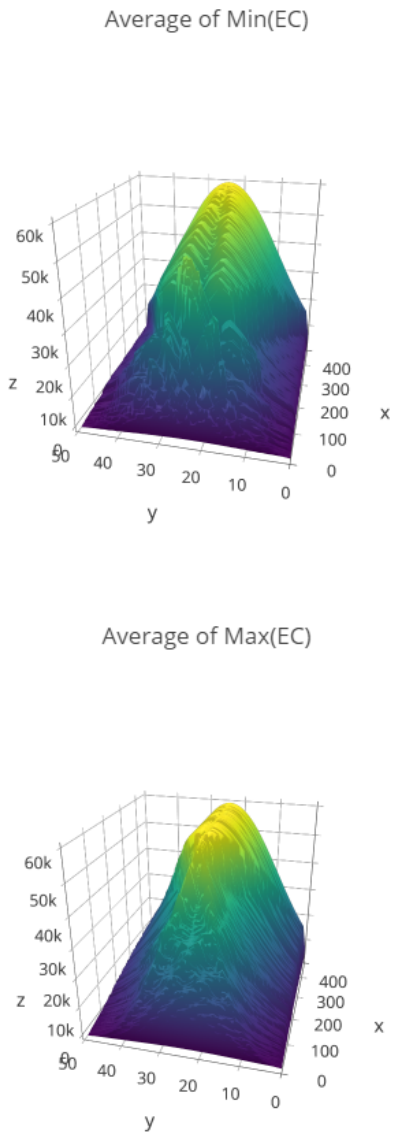

Figure A.1: Average of Min(EC)/Max(EC) for Normal Model, x: capacity, y: player ID, z: payoff, samples per game: 2000

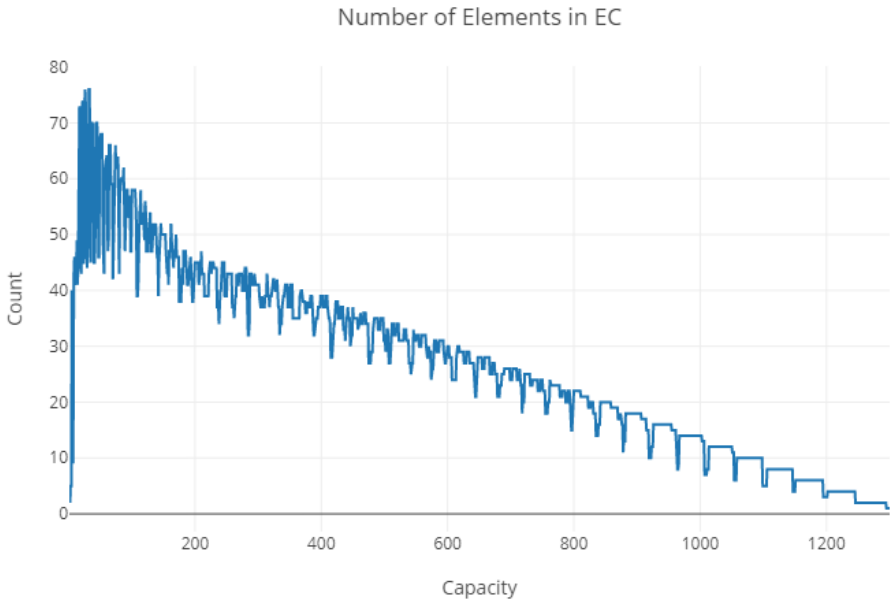

Figure A.2: Number of Elements in EC for Bottleneck Constant Model, 2000 samples per game

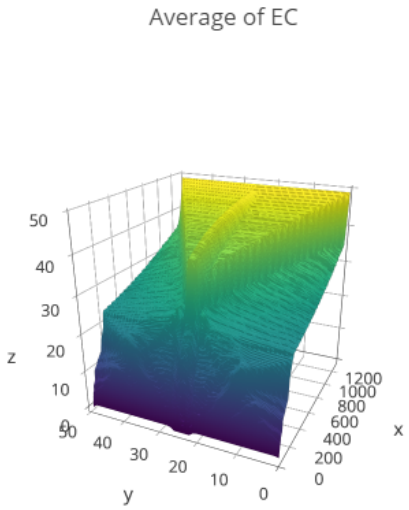

Figure A.3: Average of EC for Bottleneck Constant Model, x: capacity, y: player ID, z: payoff, samples per game: 2000

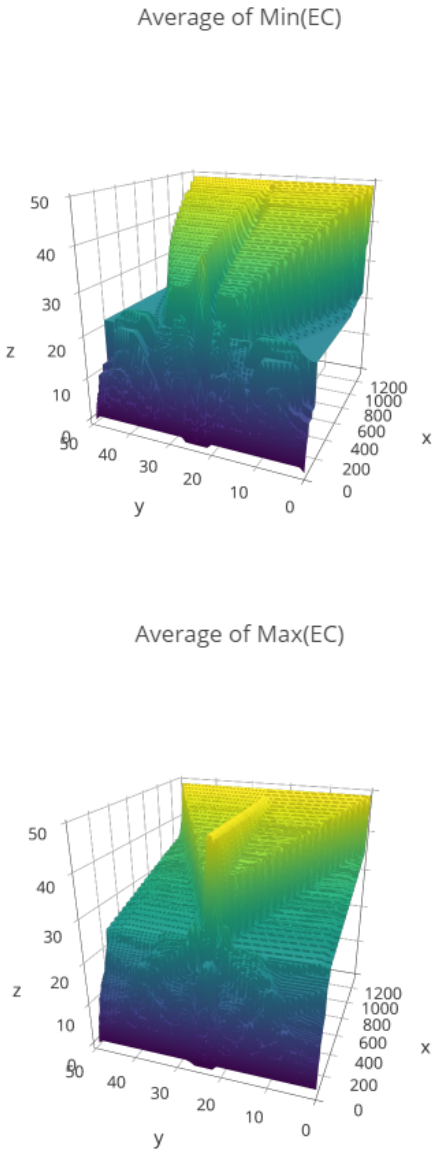

Figure A.4: Average of Min(EC)/Max(EC) for Bottleneck Constant Model, x: capacity, y: player ID, z: payoff, samples per game: 2000

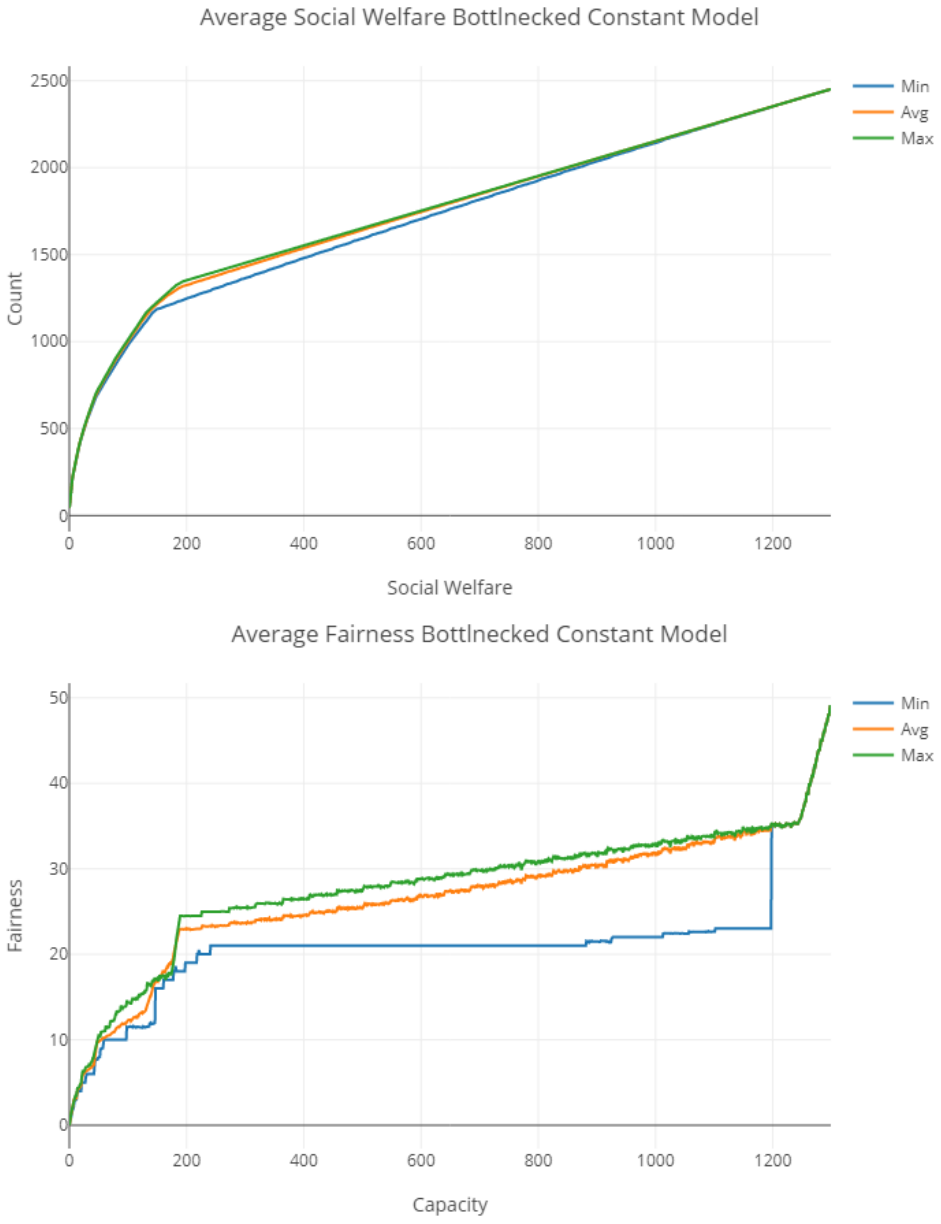

Figure A.5: Average SW and Fairness for Bottleneck Constant Model, samples per game: 2000

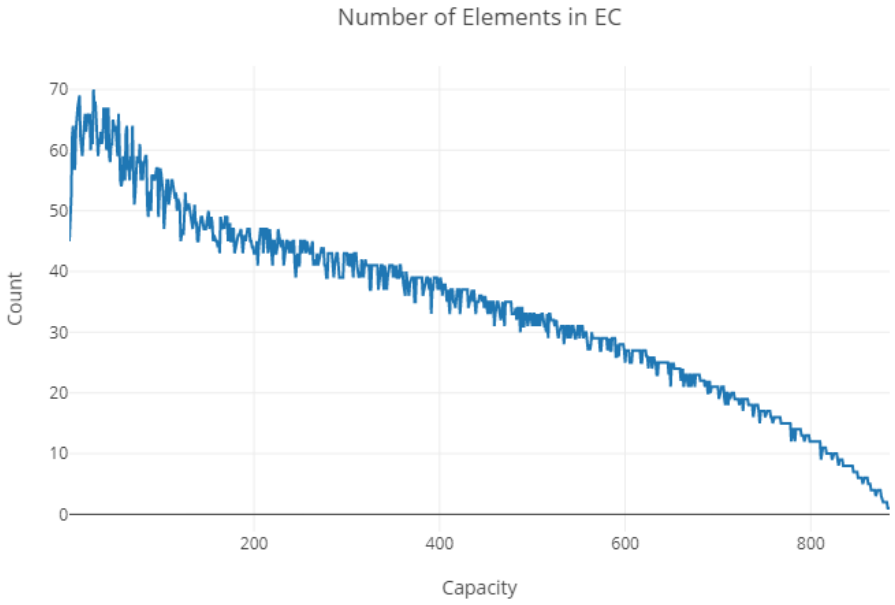

Figure A.6: Number Distinct EC for Bottleneck Normal Model, 2000 samples per game

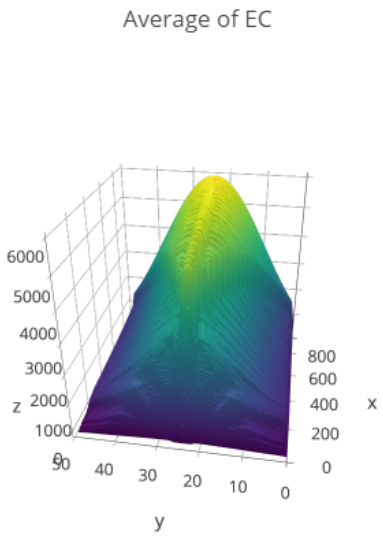

Figure A.7: Average of EC for Bottleneck Normal Model, x: capacity, y: player ID, z: payoff, samples per game: 2000

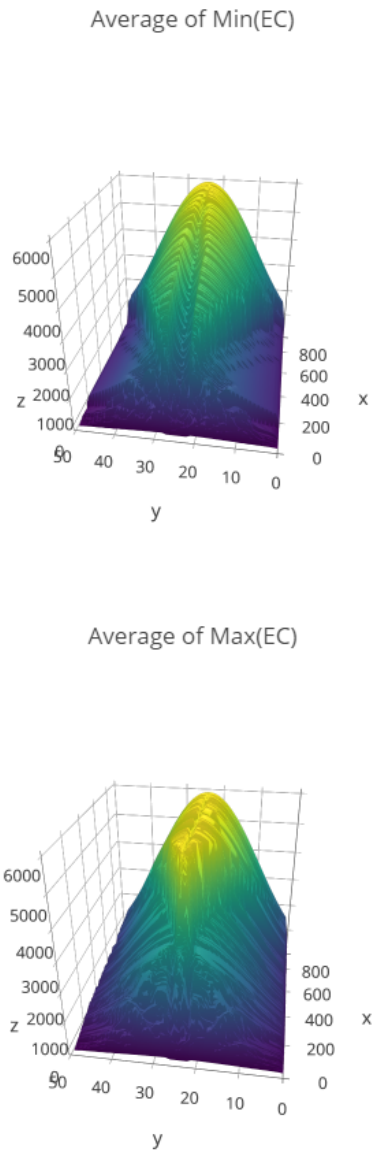

Figure A.8: Average of Min(EC)/Max(EC) for Bottleneck Normal Model, x: capacity, y: player ID, z: payoff, samples per game: 2000

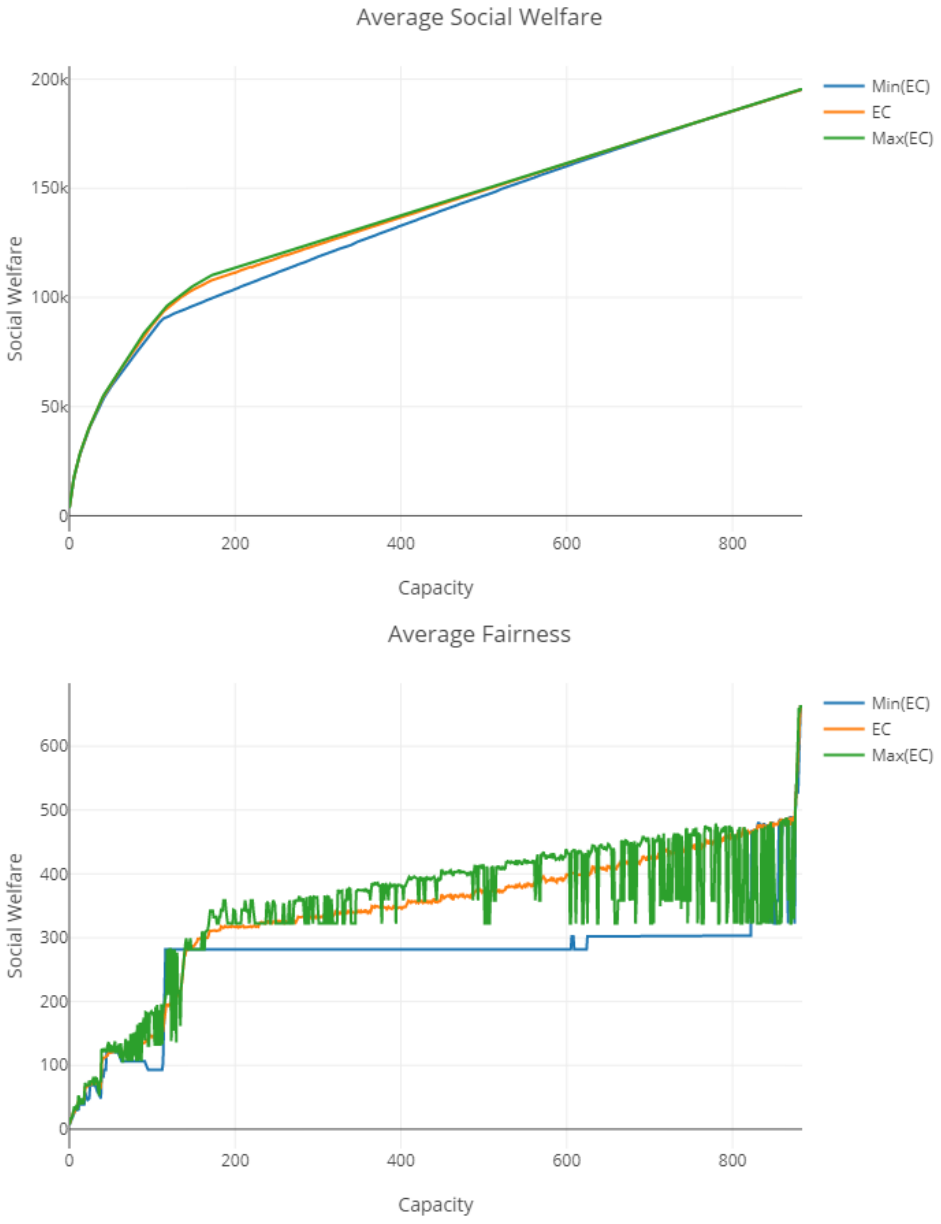

Figure A.9: Average SW and Fairness for Bottleneck Normal Model, samples per game: 2000

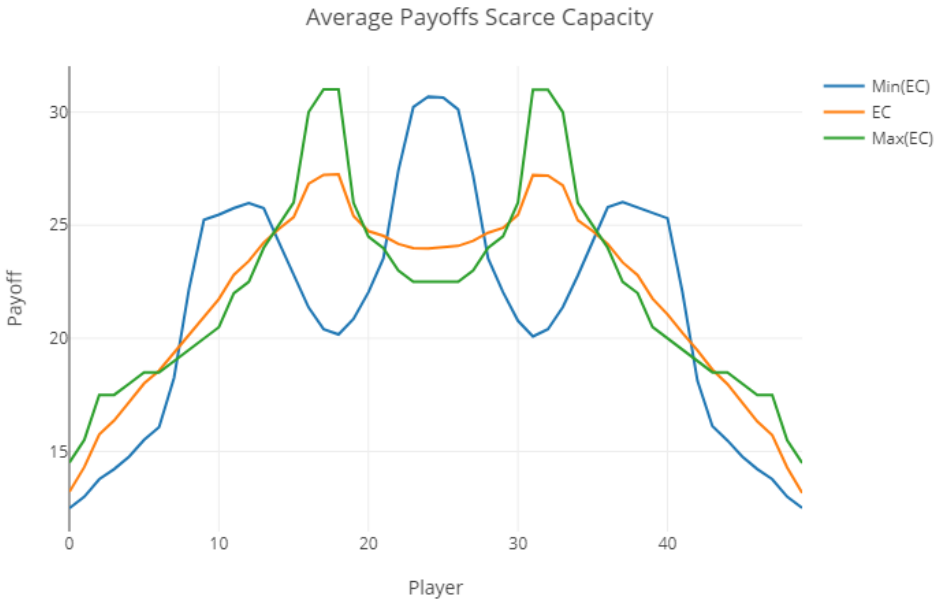

Figure A.10: Average Payoffs for Scarce Capacity in Constant Model, Samples: 1000

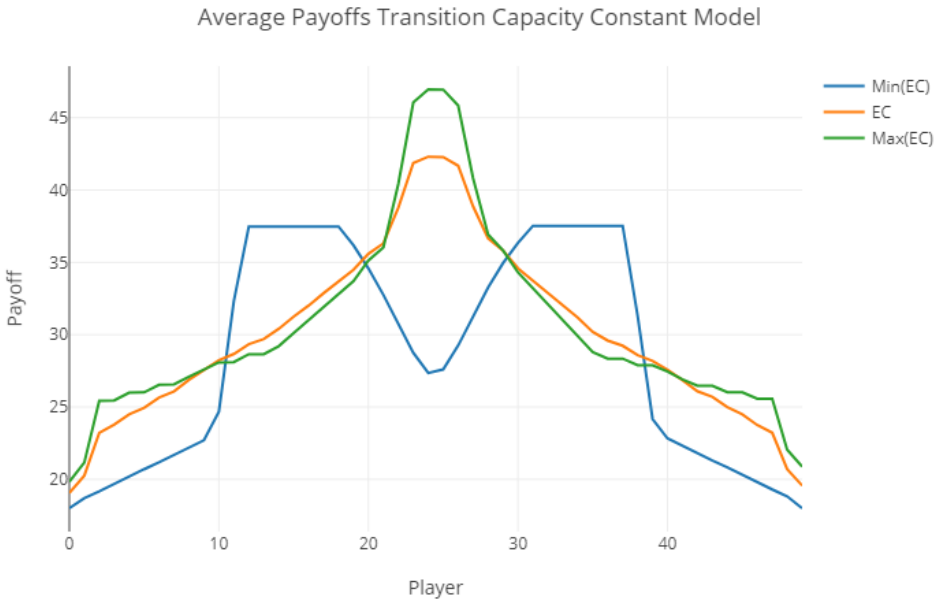

Figure A.11: Average Payoffs for Transition Capacity in Constant Model, Samples: 1000

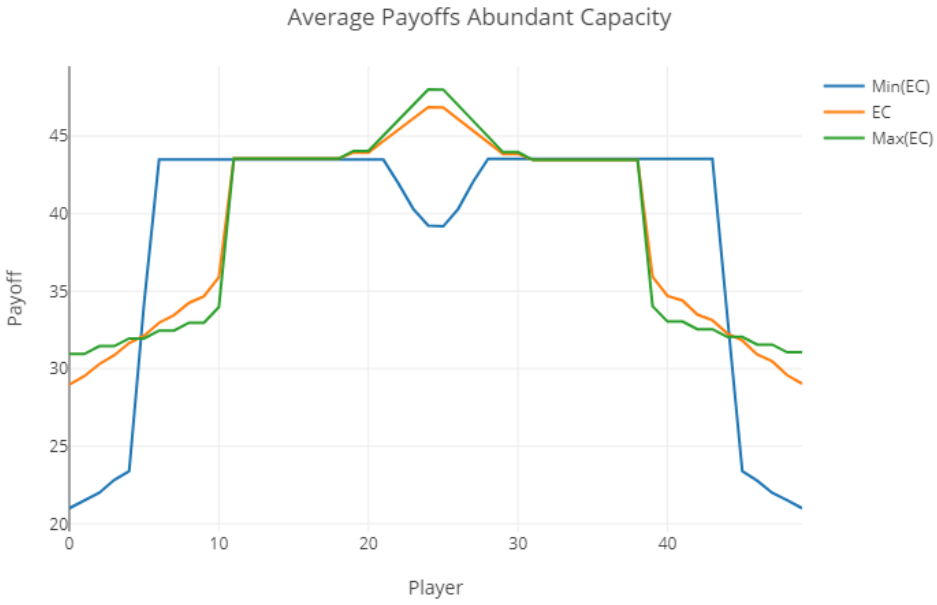

Figure A.12: Average Payoffs for Abundant Capacity in Constant Model, Samples: 1000

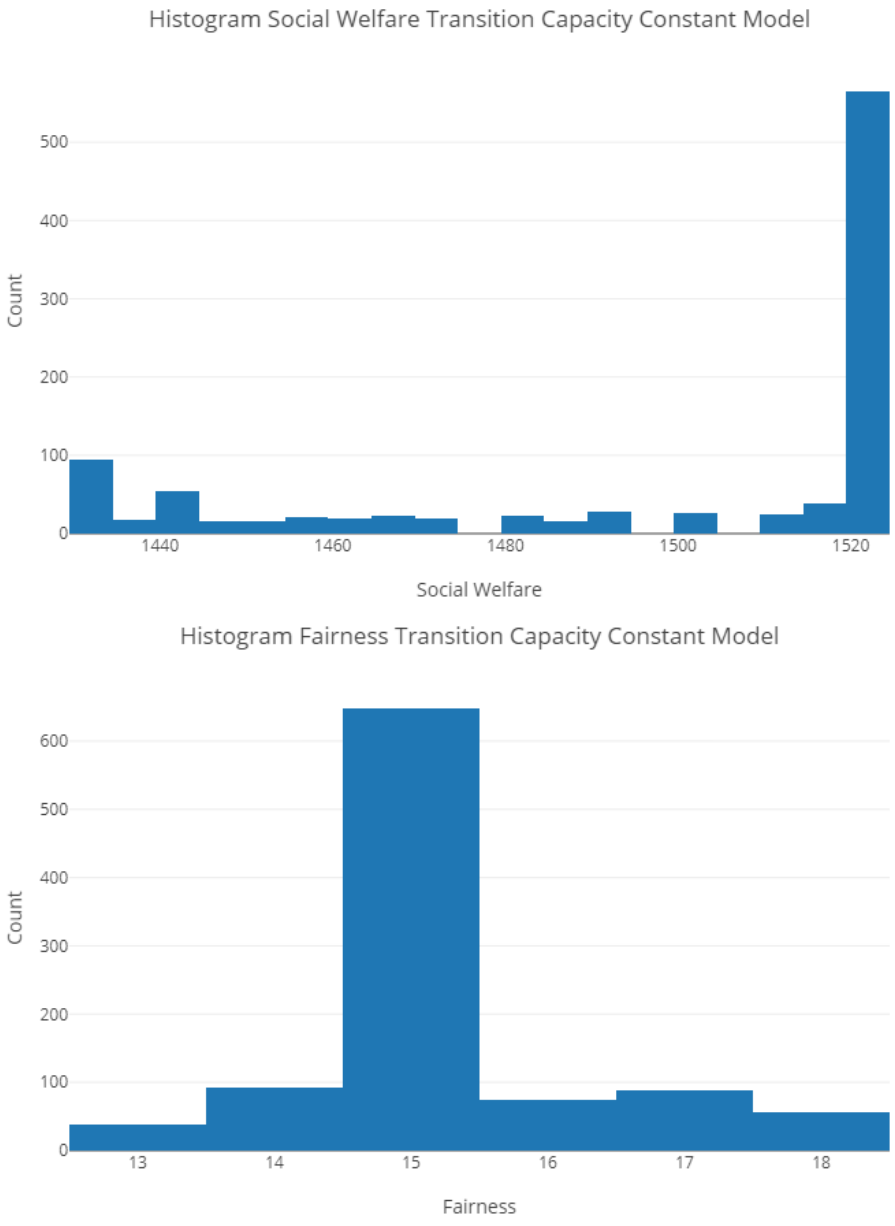

Figure A.13: Histogram of SW and Fairness Transition Capacity for Constant Model, Samples: 1000

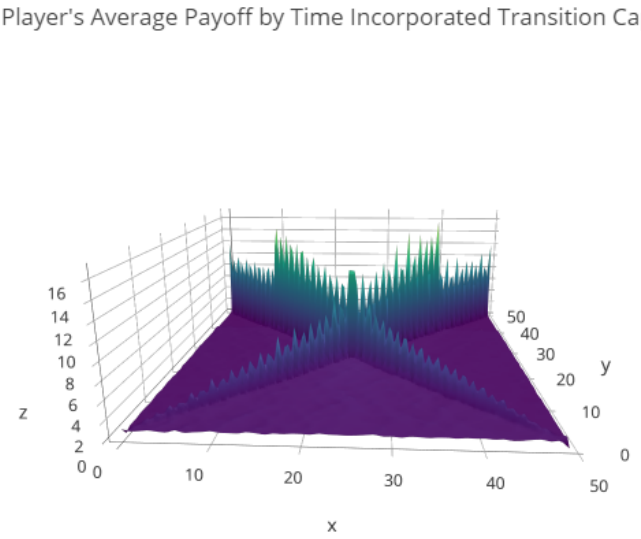

Figure A.14: Player's Average Payoff by Time Incorporated for Transition Capacity Constant Model, Samples: 5000, x: player, y: time incorporated, z: payoff

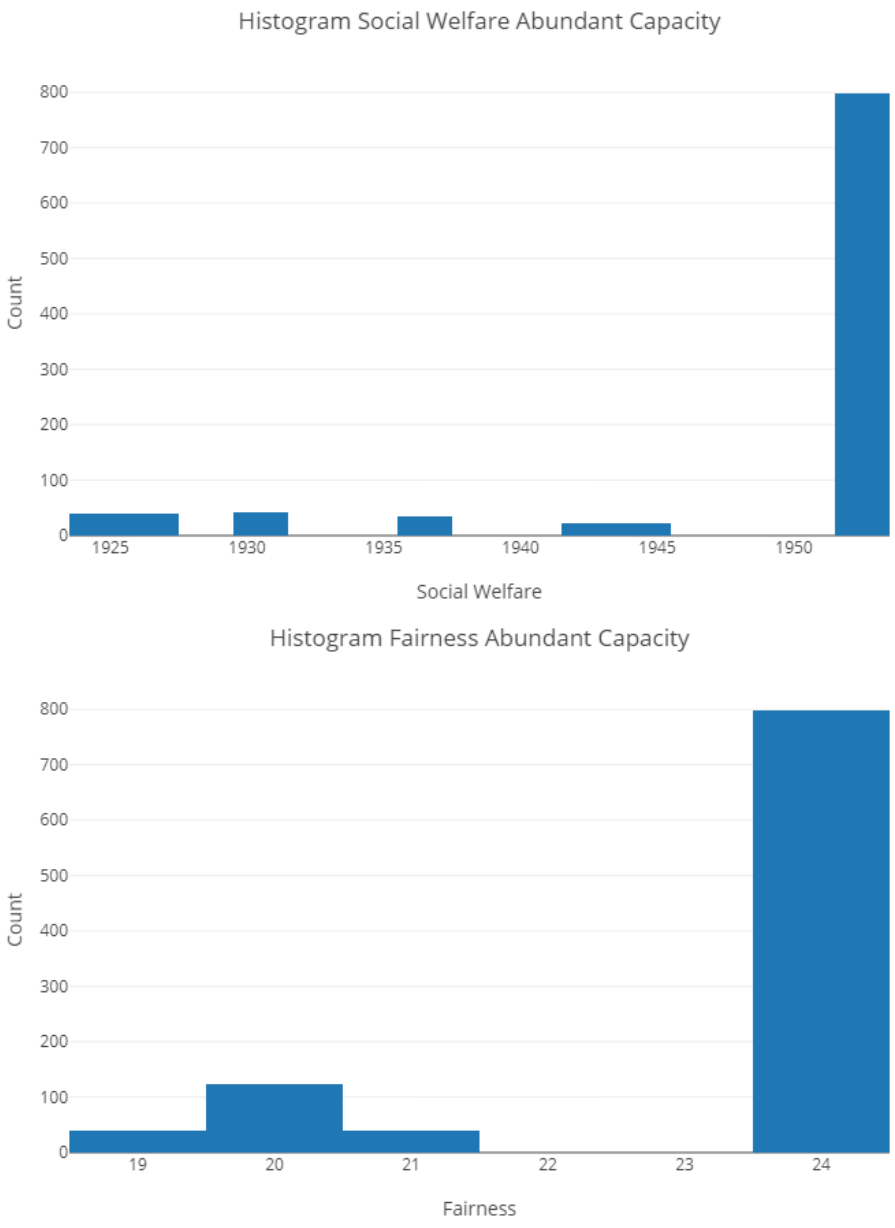

Figure A.15: Histogram of SW and Fairness for Abundant Capacity Constant Model, Samples: 1000

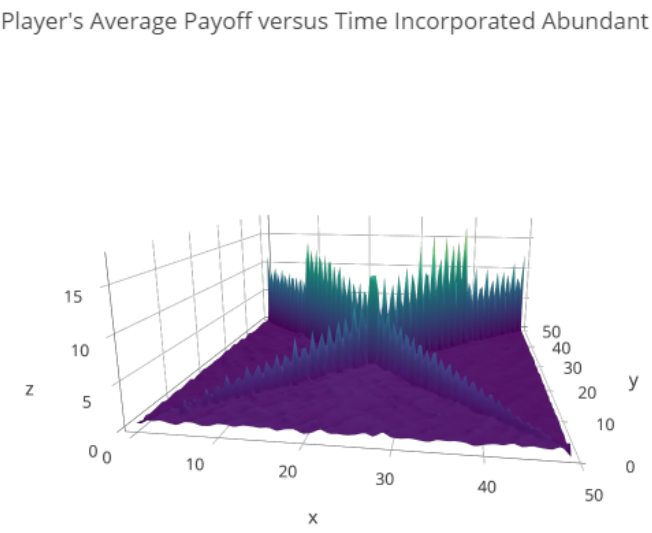

Figure A.16: Player's Average Payoff by Time Incorporated for Abundant Capacity Constant Model, Samples: 5000, x: player, y: time incorporated, z: payoff

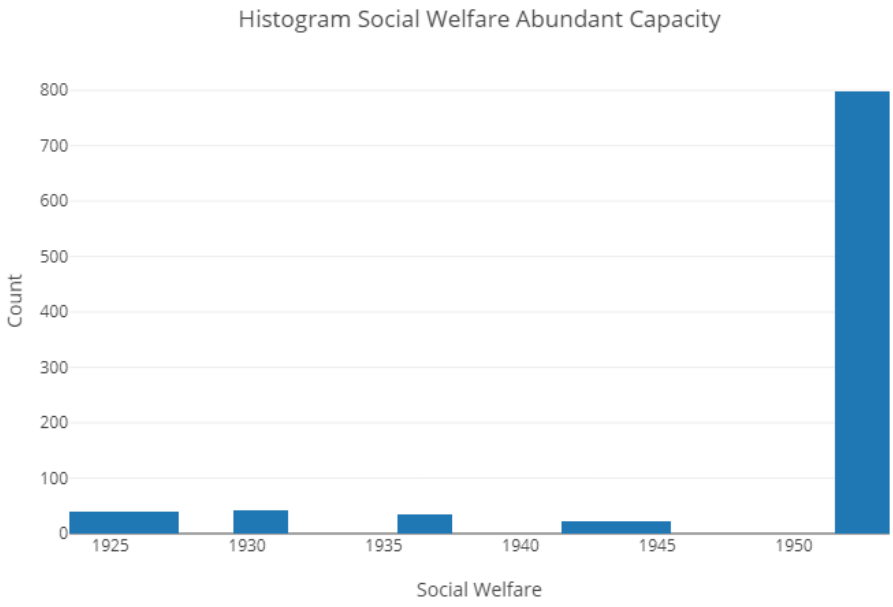

Figure A.17: Histogram of SW Random Game, Samples: 1000

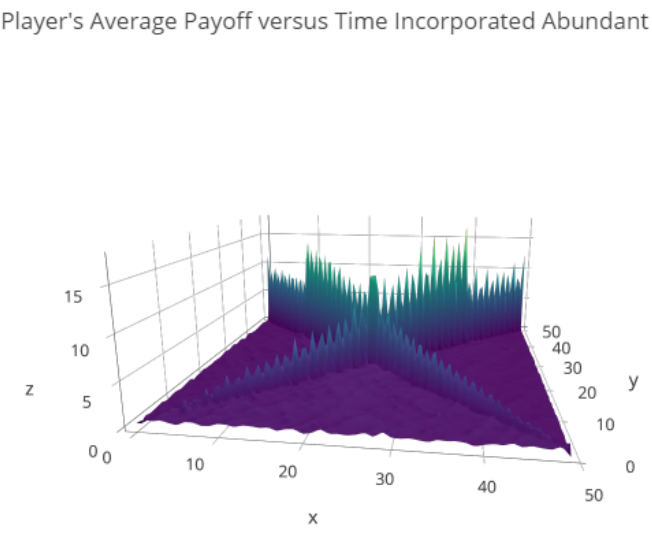

Figure A.18: Player's Average Payoff by Time Incorporated for Random Game, Samples: 1000, x: player, y: time incorporated, z: payoff