Buyer-Supplier Games: Optimization Over the Core

Nedialko B. Dimitrov [*], C. Greg Plaxton [1]

University of Texas at Austin
1 University Station C0500
Austin, Texas 78712–0233

Abstract

In a buyer-supplier game, a special type of assignment game, a distinguished player, called the buyer, wishes to purchase some combinatorial structure. A set of players, called suppliers, offer various components of the structure for sale. Any combinatorial minimization problem can be transformed into a buyer-supplier game. While most previous work has been concerned with characterizing the core of buyer-supplier games, in this paper we study optimization over the set of core vectors. We give a polynomial time algorithm for optimizing over the core of any buyer-supplier game for which the underlying minimization problem is solvable in polynomial time. In addition, we show that it is hard to determine whether a given vector belongs to the core if the base minimization problem is not solvable in polynomial time. Finally, we introduce and study the concept of focus point price, which answers the question: If we are constrained to play in equilibrium, how much can we lose by playing the wrong equilibrium?

Key words: Assignment Game, Core, Focus Point Price, Separation Oracle

1 Introduction

In this paper, we study the core of a large set of games, a subset of assignment games, which we term buyer-supplier games [3,22] [23, Chapter 6]. We are primarily concerned with efficient computations over the set of vectors

[*] Corresponding author.
Email addresses: `ned@cs.utexas.edu` (Nedialko B. Dimitrov), `plaxton@cs.utexas.edu` (C. Greg Plaxton).
[1] Supported by NSF Grants CCF–0635203 and ANI–0326001.

belonging to the core of buyer-supplier games. Before diving into an overview of buyer-supplier games, we present some connections between our work and the existing literature.

1.1 Related Work

Though suggested by Edgeworth as early as 1881 [8], the notion of the core was formalized by Gillies and Shapley [11,21], extending von Neumann and Morgenstern's work on coalitional game theory [24]. Recently, Goemans and Skutella studied the core of a cost sharing facility location game [12]. In their paper, Goemans and Skutella are primarily interested in using core vectors as a cost sharing indicator, to decide how much each customer should pay for opening the facility used by the customer. Goemans and Skutella show that, in general, the core of the cost sharing facility location game they study is empty. In contrast, for the buyer-supplier games we study, the core is always nonempty. Additionally, in our work we do not view vectors in the core as an indication of cost shares but rather as rational outcomes of negotiation amongst the players in the buyer-supplier game. Pál and Tardos extend the work of Goemans and Skutella by developing a mechanism for the cost sharing facility location game which uses the concept of an approximate core [15].

There has been great interest in comparing the game's best outcome to the best equilibrium outcome, where the term best is based on some objective function. For example, one may wish to compare the outcome maximizing the net utility for all players in the game against the best possible Nash equilibrium, with respect to net utility. Papadimitriou termed one such comparative measure as the price of anarchy [16]. Roughgarden and Tardos have studied the price of anarchy in the context of routing [18–20].

In this paper, we introduce a quantity with a similar motivation to that of the price of anarchy. Solution concepts often yield multiple predictions, or equilibria. In actual game play, however, only one of the equilibria can be chosen by the game's players. Experiments show that conditions outside the game, such as societal pressures or undue attention to a specific player, focus the players' attention on the point of a single equilibrium, which then becomes the outcome of the game. This is a common notion in game theory called the focus point. A player may receive different payoffs in different equilibria. How much is the player willing to pay for a good focus point? We define the *focus point price* with respect to a given player as the difference between the maximum and minimum equilibrium payoffs to the player. Stated succinctly, focus point price answers the question: If we are constrained to play in equilibrium, how much can we lose by playing the wrong equilibrium?

Recently, Garg et al. studied transferable utility games they call coalitional games on graphs [10]. Coalitional games on graphs are a proper subset of buyer-supplier games, which can be derived by setting the buyer's internal cost, Bcost, to zero (see Section 1.3 and Lemma 11). For some buyer-supplier games, for example the buyer-supplier facility location game, it does not appear that the game can be described with Bcost fixed to zero.

Garg et al. study the concepts of "frugality" and "agents are substitutes." They show that suppliers are substitutes if and only if the core of the game forms a lattice. In buyer-supplier games, suppliers are not always substitutes. We show, in Lemma 15, that if suppliers are substitutes, we can optimize over the core by solving a polynomially sized linear program. Garg et al. and, more recently, Karlin et al. study and characterize the frugality certain auction mechanisms; the focus point price concept introduced in this paper is quite different from frugality [13].

A third difference between Garg et al. and this work comes from the fact that, similarly to the economics literature, Garg et al. are mainly concerned with the characterization of the core: When does the core form a lattice? How do core vectors relate to auctions? We, on the other hand, are mainly concerned with characterizing optimization over the core. Our main results are in the flavor of Deng and Papadimitriou, in that we are interested in the complexity of computing using game theoretic characterizations [7].

Faigle and Kern study optimization over the core for submodular cost partition games [9]. Faigle and Kern exhibit a generic greedy-type algorithm for optimization of any linear function over the core of partition games whose value function is both submodular and *weakly increasing*, a property they define.

The greedy framework of Faigle and Kern captures certain buyer-supplier games, such as the buyer-supplier minimum spanning tree game. However, even some buyer-supplier games derived from problems that admit greedy solutions, such as the buyer-supplier shortest path game, are not amenable to the approach of Faigle and Kern. In this paper, we do not restrict ourselves to greedy algorithms. By making use of the ellipsoid method, we are able to give polynomial time algorithms for optimization over the core of any buyer-supplier game for which the underlying minimization problem is solvable in polynomial time.

To provide the reader with a simple, concrete example of optimization over the core of a buyer-supplier game, towards the end of this paper, we focus our attention on the buyer-supplier minimum spanning tree game. We give a simple greedy algorithm for this problem, which is a minor extension of Kruskal's minimum spanning tree algorithm. A greedy algorithm is provided by the work of Faigle and Kern, but their exposition involves a good deal of machinery. Our exposition is completely elementary.

Several methods, apart from buyer-supplier games, are known for transforming a combinatorial optimization problem into a game. The cores of these transformations have also been extensively studied. For example, Deng et al. show results on core non-emptyness, distinguishability of core vectors, and finding core vectors for one such transformation [6]. Caprara et al. continue the work of Deng et al. by considering a certain optimization over the set of core vectors for this alternate transformation [4].

1.2 Main Contributions

There has been increased interest from the theoretical computer science community in game theory. While problem-specific solutions may give us insight, to leverage the full power of decades of study in both research areas, we must find generic computational solutions to game theoretic problems. Indeed, others have already realized this need [1,17]. In this paper, we continue this line of work by deriving generic results for computing with core solutions in a large class of games.

The core of buyer-supplier games in the transferable utility setting is characterized by Shapley and Shubik [22]. As a minor contribution, we extend their result by showing that the core in the non-transferable utility setting is the same as the core with transferable utilities. Our primary contributions are as follows:

(1) While previous work in the economics literature has concentrated on characterizing the core of buyer-supplier games and relating core vectors to auctions, our main interest is in optimizing over the set of core vectors [3]. We provide a generally applicable algorithm, based on the ellipsoid method, for optimizing over the core. If the original minimization problem is solvable in polynomial time, we show that it is possible to optimize linear functions of core vectors in polynomial time.
(2) We fully characterize optimization over the core of buyer-supplier games by using a polynomial time reduction to show that if the original minimization problem is not solvable in polynomial time, it is impossible, in polynomial time, to test if an arbitrary vector is in the core of the buyer-supplier game.
(3) We introduce the concept of focus point price. Our positive computational results give a polynomial time algorithm for computing the buyer's focus point price in buyer-supplier games when the underlying minimization problem is solvable in polynomial time. When the underlying minimization problem is not solvable in polynomial time, we show that it

is impossible to approximate the buyer's focus point price to within *any* multiplicative factor.

1.3 Overview of Buyer-Supplier Games

The definition of a buyer-supplier game, given in Section 2.1, is self-contained and does not require an argument. However, it is also possible to transform a combinatorial minimization problem into a buyer-supplier game. Consider a combinatorial minimization problem of the following form. We have some finite set of elements $\mathcal{C}$. We designate some subsets of $\mathcal{C}$ as feasible. To capture feasibility, we use a predicate $P : 2^{\mathcal{C}} \to \{0,1\}$, where the predicate is one on all feasible subsets of $\mathcal{C}$. With each feasible set $\mathcal{A} \subseteq \mathcal{C}$, we associate a non-negative cost $f(\mathcal{A})$. The combinatorial minimization problem can then be captured by the function $\mathrm{MinProb} : 2^{\mathcal{C}} \to \Re_+$ defined by

$$\mathrm{MinProb}(\mathcal{B}) = \min_{\substack{\mathcal{A} \subseteq \mathcal{B} \\ P(\mathcal{A}) = 1}} f(\mathcal{A})$$

where $\Re_+$ denotes the non-negative real numbers.

To transform the above minimization problem into a buyer-supplier game, we associate a player with each element of $\mathcal{C}$; we call such players suppliers. We also add another player whom we call the buyer. In the game, the buyer wishes to purchase a feasible subset of $\mathcal{C}$. The suppliers, on the other hand, are offering their membership to the buyer's set at a price.

To fully specify the game's model of a realistic interaction, we let M designate the maximum investment the buyer is willing to spend on a feasible set. We decompose f such that $f(\mathcal{A}) = \mathrm{Bcost}(\mathcal{A}) + \sum_{a \in \mathcal{A}} \tau(a)$, where $\tau(a)$ is an internal cost for supplier a to be present in the buyer's set and $\mathrm{Bcost}(\mathcal{A})$ is an internal cost to the buyer for purchasing this specific feasible set. In general, many such decompositions are possible, and they produce different games. However, when specifically applying the core solution concept, Lemma 11 shows that all such decompositions are equivalent. Though it is not necessary, to remove special cases in our statements, it is convenient to let $\mathrm{Bcost}(\mathcal{A}) = M$ when $\mathcal{A} = \emptyset$ or $\mathcal{A}$ is not feasible.

Now that we have determined the internal costs for the buyer and the suppliers, we can specify the game. The buyer-supplier game is specified by the tuple $(\mathcal{C}, \tau, \mathrm{Bcost})$. The strategy set for the buyer is the power set of $\mathcal{C}$. By playing $\mathcal{A} \subseteq \mathcal{C}$, the buyer chooses to purchase the membership of the suppliers in $\mathcal{A}$. The strategy set for every supplier $a \in \mathcal{C}$ is the non-negative real numbers, indicating a bid or payment required from the buyer for the supplier's membership.

For any supplier $a \in \mathcal{C}$, we let $\beta(a)$ denote the associated bid. Let $\mathcal{A}$ be the set of suppliers chosen by the buyer. The payoff for the buyer is $M - \text{Bcost}(\mathcal{A}) - \sum_{a \in \mathcal{A}} \beta(a)$. The payoff for a supplier not in $\mathcal{A}$ is 0. The payoff for a supplier a in $\mathcal{A}$ is $\beta(a) - \tau(a)$.

Since we are applying the solution concept of the core, one may think of the game play as follows. All the players in the game sit down around a negotiating table. All the players talk amongst themselves until they reach an agreement which cannot be unilaterally and selfishly improved upon by any subset of the players. Once such an agreement is reached, game play is concluded. Since no subset of the players can unilaterally and selfishly improve upon the agreement, rationality binds the players to follow the agreement.

The fully formal definition of a buyer-supplier game is given in Section 2.1. The transformation process described above can be used to create buyer-supplier games from most combinatorial minimization problems. For example, minimum spanning tree, Steiner tree, shortest path, minimum set cover, minimum cut, single- and multi-commodity flow can all be used to instantiate a buyer-supplier game.

As a concrete example and interpretation of a buyer-supplier game, consider the buyer-supplier minimum spanning tree game. In this game, a company owns factories on every node of a graph. The company wishes to connect the factories by purchasing edges in the graph. Each edge is owned by a unique supplier player. Each supplier has an internal cost associated with the company's usage of the edge. The company has a maximum amount of money it is willing to spend on purchasing edges. Depending on the transportation conditions of a particular edge, the company may have some internal cost associated with choosing that particular edge. The buyer-supplier game paradigm yields similarly natural games when applied to other minimization problems.

In this paper we will be concerned with efficient computation over the set of core vectors. For the rest of the paper, when we say polynomial time, we mean time polynomial in the size of the parameter $\mathcal{C}$, which is also polynomial in the number of players of the buyer-supplier game.

1.4 Organization of the Paper

In Section 2 we define buyer-supplier games and the core of a game. In Section 3 we characterize the core of buyer-supplier games. In Section 4 we give positive computational results, namely the generic algorithm for optimizing over the set of core vectors. In Section 5 we give negative computational results by showing polynomial time equivalence between several related problems. In Section 6 we give the problem-specific combinatorial algorithm for the buyer-supplier game arising from the minimum spanning tree problem.

2 Definitions

In this section, we formally define buyer-supplier games and give the game theoretic definitions required for our analysis.

2.1 Buyer-Supplier Games

Let $\mathcal{C}$ be a finite set and M be a non-negative real number. Let τ be a function from $\mathcal{C}$ to $\Re_+$. Let Bcost be a function from $2^{\mathcal{C}}$ to $\Re_+$ such that $\text{Bcost}(\emptyset) = M$. The simplifying condition that $\text{Bcost}(\emptyset) = M$ is not required. We explain the condition's purpose later in this section. For $\mathcal{A} \subseteq \mathcal{C}$, let $\text{Eval}(\tau, \text{Bcost}, \mathcal{A})$ denote $\text{Bcost}(\mathcal{A}) + \sum_{a \in \mathcal{A}} \tau(a)$. For $\mathcal{A} \subseteq \mathcal{C}$, let $\text{MinEval}(\tau, \text{Bcost}, \mathcal{A})$ denote $\min_{\mathcal{B} \subseteq \mathcal{A}} \text{Eval}(\tau, \text{Bcost}, \mathcal{B})$. We will omit the parameters τ and Bcost from the functions $\text{Eval}(\tau, \text{Bcost}, \mathcal{A})$ and $\text{MinEval}(\tau, \text{Bcost}, \mathcal{A})$ when the value is clear.

Given a tuple $(\mathcal{C}, \tau, \text{Bcost})$, we proceed to define a buyer-supplier game. Associate a player with each element of $\mathcal{C}$. Call the players in $\mathcal{C}$ suppliers. Let there also be another player, μ, whom we call the buyer. Let $\mathcal{P} = \mathcal{C} \cup \{\mu\}$ be the set of players for the buyer-supplier game.

The strategy for supplier a is a tuple $(\beta(a), p_a)$ with $\beta(a) \in \Re_+$ and $p_a : \mathcal{P} \to \Re_+$. The first element, $\beta(a)$, represents supplier a's bid to the buyer, requiring the buyer to pay $\beta(a)$ for using the supplier's services. The second element, p_a, represents the non-negative side payments supplier a chooses to make to the game's players. By $p_a(b)$ we denote the side payment a makes to player b.

The strategy for the buyer, μ, is a tuple $(\mathcal{A}, p_\mu)$ where $\mathcal{A} \subseteq \mathcal{C}$ and $p_\mu : \mathcal{P} \to \Re_+$. The first element, $\mathcal{A}$, represents the suppliers chosen by the buyer for a purchase. Similarly to a supplier, the second element, p_μ, represents the non-negative side payments the buyer chooses to make to the game's players.

For each player $a \in \mathcal{P}$ we denote the player's strategy set by $\mathcal{S}_a$. For a set of players $\mathcal{A} \subseteq \mathcal{P}$, we denote the set of strategies $\bigotimes_{a \in \mathcal{A}} \mathcal{S}_a$ by $\mathcal{S}_{\mathcal{A}}$. We call elements of $\mathcal{S}_{\mathcal{A}}$ strategy vectors. We index strategy vectors from $\mathcal{S}_{\mathcal{A}}$ by the elements of $\mathcal{A}$.

We now define the utility function for each player. Suppose strategy $s \in \mathcal{S}_{\mathcal{P}}$ is played. Specifically, suppose that $(\mathcal{A}, p_\mu) \in \mathcal{S}_\mu$ and $(\beta(a), p_a) \in \mathcal{S}_a$ for each

$a \in \mathcal{C}$ are played. The utility function for buyer is $u_\mu(s) = M - [\text{Bcost}(\mathcal{A}) + \sum_{a \in \mathcal{A}} \beta(a)] + [\sum_{b \in \mathcal{P}} p_b(\mu) - \sum_{b \in \mathcal{P}} p_\mu(b)]$. The utility for a supplier a in $\mathcal{A}$ is $u_a(s) = \beta(a) - \tau(a) + [\sum_{b \in \mathcal{P}} p_b(a) - \sum_{b \in \mathcal{P}} p_a(b)]$. The utility for a supplier a not in $\mathcal{A}$ is $u_a(s) = [\sum_{b \in \mathcal{P}} p_b(a) - \sum_{b \in \mathcal{P}} p_a(b)]$.

Interpreting, the buyer begins with a total of M utility and chooses to make a purchase from each supplier in $\mathcal{A}$. The buyer gives $\beta(a)$ to each supplier $a \in \mathcal{A}$ and loses an extra $\text{Bcost}(\mathcal{A})$ from the initial M utility. Each supplier a in $\mathcal{A}$ receives the bid payment from the buyer and loses $\tau(a)$ because the supplier must perform services for the buyer. The distribution of sidepayments completes the utility functions. The requirement that $\text{Bcost}(\emptyset) = M$ lets the strategy $\emptyset$ stand as a "don't play" strategy for the buyer. To remove the requirement, we could introduce a specific "don't play" strategy to the buyer's strategy set, however this creates a special case in most of our proofs.

Let the sidepayment game we have defined be denoted SP. Let NOSP denote the same game with the additional requirement that all sidepayments be fixed to zero. In other words, in NOSP we restrict the strategy set for each $a \in \mathcal{P}$ so that p_a is identically zero.

2.2 Game Theoretic Definitions

All of the definitions in this section closely follow those of Shubik [23, Chapter 6].

We call a vector in $\Re^{|\mathcal{P}|}$, indexed by $a \in \mathcal{P}$, a *payoff vector.* We say a payoff vector π is *realized* by a strategy vector $s \in \mathcal{S}_\mathcal{P}$ if $\pi_a = u_a(s)$ for all $a \in \mathcal{P}$.

Let π be a payoff vector and s be a strategy vector in $\mathcal{S}_\mathcal{A}$ for $\mathcal{A} \subseteq \mathcal{P}$. Let t be any strategy vector in $\mathcal{S}_\mathcal{P}$ such that the restriction of t to the coordinates in $\mathcal{A}$ is equal to s. If for all t and for all $a \in \mathcal{A}$ we have $\pi_a \le u_a(t)$, we say that the players in $\mathcal{A}$ can *guarantee* themselves payoffs of at least π by playing s.

We use Shubik's alpha theory to define our characteristic sets [23, pp. 134-136]. Thus for a set of players $\mathcal{A} \subseteq \mathcal{P}$, we define the characteristic set, $V(\mathcal{A})$, to be the set of all payoff vectors π such that there is a strategy vector $s \in \mathcal{S}_\mathcal{A}$, possibly dependent on π, with which the players in $\mathcal{A}$ can guarantee themselves payoffs of at least π. In the transferable utility setting, SP, the characteristic sets can be replaced with a characteristic function. Given the definitions of the utility functions in Section 2.1, the characteristic function $\tilde{V}(\mathcal{A})$ for a set of players $\mathcal{A}$ is equal to $M - \text{MinEval}(\tau, \text{Bcost}, \mathcal{A} - \{\mu\})$.

We say that a set $\mathcal{A} \subseteq \mathcal{P}$ of *players are substitutes* if $\tilde{V}(\mathcal{P}) - \tilde{V}(\mathcal{P} - \mathcal{B}) \ge \sum_{a \in \mathcal{B}} \tilde{V}(\mathcal{P}) - \tilde{V}(\mathcal{P} - \{a\})$ for all $\mathcal{B} \subseteq \mathcal{A}$.

We say that a payoff vector π dominates a payoff vector ν through a set $\mathcal{A} \subseteq \mathcal{P}$ if $\pi_a > \nu_a$ for all $a \in \mathcal{A}$. In other words, π dominates ν through $\mathcal{A}$ when each player in $\mathcal{A}$ does better in π than in ν.

For a set of players $\mathcal{A} \subseteq \mathcal{P}$, we define $D(\mathcal{A})$ as the set of all payoff vectors which are dominated through $\mathcal{A}$ by a payoff vector in $V(\mathcal{A})$. Interpreting, the players in $\mathcal{A}$ would never settle for a payoff vector $\pi \in D(\mathcal{A})$ since they can guarantee themselves higher payoffs than those offered in π.

The *core* of a game consists of all $\pi \in V(\mathcal{P})$ such that $\pi \notin D(\mathcal{A})$ for all $\mathcal{A} \subseteq \mathcal{P}$.

3 A Characterization of the Core

The characterization of the core of buyer-supplier games in the transferable utility setting was done by Shapley and Shubik [22]. In this section, we show the surprising result that the same characterization holds in the non-transferable utility setting. In general, it is not the case that the core of the transferable utility and non-transferable utility versions of a game are the same. For example, the buyer may be able to use bribes to alter the bidding strategies of some suppliers, and thus reduce the bids of other suppliers. The following condition characterizes the core of buyer-supplier games. A payoff vector π is in the core of a buyer-supplier game defined by $(\mathcal{C}, \tau, \text{Bcost})$ if and only if it satisfies

$$\begin{aligned} \pi_a &\geq 0 && \forall a \in \mathcal{P}, \\ \sum_{a \in \mathcal{A}} \pi_a &\leq \text{MinEval}(\tau, \text{Bcost}, \mathcal{C} - \mathcal{A}) - \text{MinEval}(\tau, \text{Bcost}, \mathcal{C}) && \forall \mathcal{A} \subseteq \mathcal{C}, \\ \pi_\mu &= M - \text{MinEval}(\tau, \text{Bcost}, \mathcal{C}) - \sum_{a \in \mathcal{C}} \pi_a. \end{aligned}$$

As many of the proofs required to show the result are straight forward, to simplify the presentation we present statements of key lemmas and a few selected proofs. The main idea that gives the result, captured in Lemma 1, is that sidepayments can be simulated through altering the bids of the suppliers to the buyer. In Section 3.1, we give some preliminary lemmas for the games NOSP and SP. In Section 3.2, we show that the core of SP is the same as the core of NOSP and give a characterization of the core.

3.1 Preliminary Lemmas on the Core of NOSP *and* SP

This section contains some preliminary lemmas that are useful in characterizing the core of NOSP and SP.

For some of our proofs it is convenient to think of SP as a two stage distribution of wealth, where the strategy $s \in \mathcal{S}_{\mathcal{P}}$ determines the utility transfers. Initially, the buyer has M utility and all suppliers have zero utility. In the first stage, the buyer gives $\beta(b)$ to each supplier $b \in \mathcal{A}$ and loses an extra $\text{Bcost}(\mathcal{A})$ from the initial M utility. Also in the first stage, each supplier $b \in \mathcal{A}$ loses $\tau(b)$ of utility. In the second stage, side payments are distributed.

Lemma 1 *Let $s \in \mathcal{S}_{\mathcal{A}\cup\mu}$ be such that $s_\mu = (\mathcal{A}, p_\mu)$. If s guarantees the players in $\mathcal{A} \cup \mu$ payoffs of at least $\pi \in \Re^{|\mathcal{A}\cup\mu|}$, then there is a $t \in \mathcal{S}_{\mathcal{A}\cup\mu}$ such that*

- *All side payments from players in $\mathcal{A} \cup \mu$ to players in $\mathcal{A} \cup \mu$ are fixed to zero in t*
- $t_\mu = (\mathcal{A}, 0)$
- *t also guarantees payoffs of at least π.*

PROOF. We show how to sequentially remove the specified side payments while maintaining the payoff guarantee.

Let a and b be suppliers in $\mathcal{A}$. Let $s_a = (\beta(a), p_a)$ and $s_b = (\beta(b), p_b)$.

First, consider a supplier to supplier payment. Suppose that $p_a(b) = \lambda$, that is, supplier a pays λ to supplier b. Because both a and b are in $\mathcal{A}$, we can achieve the same utility transfer as the side payment by setting the side payment to zero and changing $\beta(a)$ to $\beta(a) - \lambda$ and $\beta(b)$ to $\beta(b) + \lambda$. Thus, we can zero out the side payment from a to b.

Now, consider a supplier to buyer payment. Suppose that $p_a(\mu) = \lambda$. In other words supplier a pays λ to the buyer. We can achieve the same utility transfer as the side payment by setting the side payment to zero and changing $\beta(a)$ to $\beta(a) - \lambda$. Thus, we can zero out the side payment from a to μ.

A similar change works for a payment from the buyer to a supplier.

Lemma 2 *Let strategy vector $s \in \mathcal{S}_{\mathcal{P}}$ realize payoff vector π. If $s_\mu = (\mathcal{A}, p_\mu)$ and there exists $a \in \mathcal{C} - \mathcal{A}$ such that $\pi_a > 0$, then $\pi \in D(\mathcal{A} \cup \mu)$ in both* SP *and* NOSP.

Lemma 3 *If π is in the core of* SP *or* NOSP, *then $\pi_a \geq 0$ for all $a \in \mathcal{P}$.*

Lemma 4 *Let π be a payoff vector, and let s be a strategy vector in $\mathcal{S}_{\mathcal{P}}$. If the players in $\mathcal{P}$ can guarantee themselves payoffs of at least π by playing s, but s does not realize π, then π is not in the core of either* SP *or* NOSP*.*

3.2 Core Equivalence and Characterization

In this section, we prove that the core of NOSP is the same as the core of SP and give a characterization of the core. All the lemmas in this section are used solely to prove the main result of the section, Theorems 9 and 10.

Lemma 5 *Let π be a payoff vector. If $\pi \in D(\mathcal{A})$ in* NOSP*, then $\pi \in D(\mathcal{A})$ in* SP*.*

PROOF. The players in $\mathcal{A}$ can follow exactly the same strategy in SP as they would in NOSP.

Lemma 6 *Let π be a payoff vector. If $\pi \in V(\mathcal{P})$ in* SP *and π is in the core of* SP*, then $\pi \in V(\mathcal{P})$ in* NOSP*.*

PROOF. By Lemma 4, π is realized by some strategy vector $s \in \mathcal{S}_{\mathcal{P}}$ in SP. The result follows by Lemma 1.

Lemma 7 *Let π be a payoff vector. If $\pi \in V(\mathcal{P})$ in* NOSP*, then $\pi \in V(\mathcal{P})$ in* SP*.*

PROOF. By the definition of $V(\mathcal{P})$, the players in $\mathcal{P}$ can guarantee themselves payoffs of at least π by playing some strategy $s \in \mathcal{S}_{\mathcal{P}}$ in NOSP. The players in $\mathcal{P}$ can follow exactly the same strategy in SP.

Lemma 8 *If payoff vector π is in the core of* NOSP*, then for all $\mathcal{A} \subseteq \mathcal{P}$ we have $\pi \notin D(\mathcal{A})$ in* SP*.*

PROOF. We use a proof by contradiction. Suppose that $\pi \in D(\mathcal{A})$ in SP for some $\mathcal{A} \subseteq \mathcal{P}$. Thus, there is a strategy vector $s \in \mathcal{S}_{\mathcal{A}}$ in SP which guarantees each player in $\mathcal{A}$ a greater payoff than the payoff given in π.

Since π is in the core of NOSP, by Lemma 3 we know $\pi_a \geq 0$ for all $a \in \mathcal{A}$.

We split the proof into two cases. In the first case, suppose that $\mu \notin \mathcal{A}$. It is impossible for s to guarantee a payoff greater than 0 for any player in $\mathcal{A}$ since the buyer can always play $\emptyset$. Thus, we get a contradiction with the assumption that $\pi \in D(\mathcal{A})$ in SP.

For the second case, suppose $\mu \in \mathcal{A}$. Let $s_\mu = (\mathcal{B}, p_\mu)$. There can not be some supplier in $\mathcal{B}$ but not in $\mathcal{A}$ since that supplier can always play the strategy $(\lambda, 0)$ where $\lambda > M$ to give the buyer a negative payoff. Since $\pi_\mu \geq 0$, the existence of a supplier in $\mathcal{B} - \mathcal{A}$ contradicts the assumption that $\pi \in D(\mathcal{A})$ in SP.

Thus, $\mathcal{B} \subseteq \mathcal{A} - \{\mu\}$.

If there is some supplier a in $\mathcal{A}$ but not in $\mathcal{B}$, since $\pi_a \geq 0$, we know that s must guarantee a payoff greater than 0 for a. Let ν be the payoff vector realized when the players in $\mathcal{A}$ follow s and each of the players in $\mathcal{P} - \mathcal{A}$ follow the strategy $(0, 0)$. Thus, $\nu_a > 0$ and $\nu_b > \pi_b$ for all $b \in \mathcal{A}$. By Lemma 2, we have $\nu \in D(\mathcal{B} \cup \mu)$ in SP. Since $\nu \in D(\mathcal{B} \cup \mu)$ in SP, $\nu_b > \pi_b$ for all $b \in \mathcal{A}$, and $\mathcal{B} \subseteq \mathcal{A} - \{\mu\}$, we have $\pi \in D(\mathcal{B} \cup \mu)$ in SP.

Thus, $s_\mu = (\mathcal{B}, p_\mu)$ and $\pi \in D(\mathcal{B} \cup \mu)$ in SP. By Lemma 1, we also have $\pi \in D(\mathcal{B} \cup \mu)$ in NOSP, which contradicts the fact that π is in the core of NOSP.

Theorem 9 *The core of* NOSP *is equal to the core of* SP*.*

PROOF. By Lemma 6 and the contrapositive of Lemma 5, if payoff vector π is in the core of SP, then π is in the core of NOSP. By Lemmas 7 and 8, if payoff vector π is in the core of NOSP, then π is in the core of SP.

The following characterization of the core of SP follows from the work of Shapley and Shubik [22]. By the preceding theorem, the same characterization is true of NOSP. In the statement of the characterization theorem, the parameter τ is made explicit, though its value is clear from the definition of the buyer-supplier game.

Theorem 10 *A payoff vector π is in the core of* SP *(or* NOSP*) if and only if it satisfies*

$$\pi_a \geq 0 \qquad \text{for all } a \in \mathcal{P} \qquad (1)$$

$$\sum_{a \in \mathcal{A}} \pi_a \leq \text{MinEval}(\tau, \mathcal{C} - \mathcal{A}) - \text{MinEval}(\tau, \mathcal{C}) \qquad \text{for all } \mathcal{A} \subseteq \mathcal{C} \qquad (2)$$

$$\pi_\mu = M - \text{MinEval}(\tau, \mathcal{C}) - \sum_{a \in \mathcal{C}} \pi_a \qquad (3)$$

Lemma 11 *Let* $\text{Bcost}^*(\mathcal{A}) = \sum_{a \in \mathcal{A}} \tau(a) + \text{Bcost}(\mathcal{A})$*. The core of the buyer-supplier games defined by* $(\mathcal{C}, \tau, \text{Bcost})$ *and* $(\mathcal{C}, 0, \text{Bcost}^*)$ *is the same.*

PROOF. We have $\text{Eval}(\tau, \text{Bcost}, \mathcal{B}) = \text{Eval}(0, \text{Bcost}^*, \mathcal{B})$ for all $\mathcal{B} \subseteq \mathcal{C}$ by the definition of Eval and Bcost*. Thus, we have $\text{MinEval}(\tau, \text{Bcost}, \mathcal{A}) = \text{MinEval}(0, \text{Bcost}^*, \mathcal{A})$ for all $\mathcal{A} \subseteq \mathcal{C}$. The result follows from Theorem 10.

4 Polynomial Time Optimization Over the Core Vectors

We define the separation problem on a set of linear inequalities $\mathcal{A}$ as follows. Given a vector π, if π satisfies all of the inequalities in $\mathcal{A}$, then do nothing; otherwise, output a violated inequality $a \in \mathcal{A}$. It is well known that the separation problem is polynomial time equivalent to linear function optimization over the same set of inequalities [14, p. 161].

Let $(\mathcal{C}, \tau, \text{Bcost})$ define a buyer-supplier game. In this section, to simplify the notation, we will omit the parameter Bcost from Eval and MinEval since it is fixed by the buyer-supplier game.

In this section, we will analyze an algorithm to solve the separation problem for the exponentially sized set of inequalities given in Equations (1), (2), and (3). We now give the algorithm, which we call the separation algorithm. Given the payoff vector π as input,

1 Iterate over Equations (1) and (3) to check that they hold. If some equation does not hold, output that equation and halt.
2 Compute $\mathcal{F} \subseteq \mathcal{C}$ such that $\text{Eval}(\tau, \mathcal{F}) = \text{MinEval}(\tau, \mathcal{C})$. If there is some $a \in \mathcal{C} - \mathcal{F}$ with $\pi_a > 0$, output the inequality from Equation (2) corresponding to $\{a\}$ and halt.
3 Define $\hat{\tau}(a) = \tau(a) + \pi_a$ for $a \in \mathcal{C}$. Now, compute $\hat{\mathcal{F}} \subseteq \mathcal{C}$ such that $\text{Eval}(\hat{\tau}, \hat{\mathcal{F}}) = \text{MinEval}(\hat{\tau}, \mathcal{C})$. If $\text{Eval}(\hat{\tau}, \hat{\mathcal{F}}) < \text{Eval}(\hat{\tau}, \mathcal{F})$, output the inequality from Equation (2) corresponding to $\mathcal{F} - \hat{\mathcal{F}}$. Otherwise, halt.

Theorem 12 *If given an input $\hat{\tau} : \mathcal{C} \to \Re_+$ it is possible to compute both* $\text{Eval}(\hat{\tau}, \mathcal{A})$*, for any $\mathcal{A} \subseteq \mathcal{C}$, and $\mathcal{F} \subseteq \mathcal{C}$ such that* $\text{Eval}(\hat{\tau}, \mathcal{F}) = \text{MinEval}(\hat{\tau}, \mathcal{C})$ *in polynomial time, then the separation problem for Equations (1), (2), and (3) is solvable in polynomial time. Furthermore, optimizing any linear function of π over Equations (1), (2), and (3) is also possible in polynomial time.*

PROOF. It is clear that given the theorem's assumptions, the separation algorithm runs in polynomial time. The statement follows from Lemmas 13 and 14. The final claim of the theorem follows from the equivalence of separation and optimization.

Lemma 13 *If the separation algorithm returns an inequality on input π, then π violates the returned inequality.*

PROOF. If the algorithm returns an inequality in step 1, then the inequality is violated since the algorithm performed a direct check.

If the algorithm returns an inequality in step 2, then the inequality is violated since $\pi_a > 0$, but $\text{MinEval}(\tau, \mathcal{C} - \{a\}) = \text{MinEval}(\tau, \mathcal{C}) = \text{Eval}(\tau, \mathcal{F})$.

Suppose the algorithm returns an inequality in step 3. Thus, $\text{Eval}(\hat{\tau}, \hat{\mathcal{F}}) < \text{Eval}(\hat{\tau}, \mathcal{F})$. Applying the definitions of Eval and $\hat{\tau}$, $\sum_{a \in \hat{\mathcal{F}}} \pi_a + \text{Eval}(\tau, \hat{\mathcal{F}}) < \sum_{a \in \mathcal{F}} \pi_a + \text{Eval}(\tau, \mathcal{F})$.

Since the algorithm reaches step 3, we know that $\pi_a = 0$ for all $a \in \mathcal{C} - \mathcal{F}$. Thus, we have $\sum_{a \in \hat{\mathcal{F}} \cap \mathcal{F}} \pi_a + \text{Eval}(\tau, \hat{\mathcal{F}}) < \sum_{a \in \mathcal{F}} \pi_a + \text{Eval}(\tau, \mathcal{F})$, which in turn gives $\text{Eval}(\tau, \hat{\mathcal{F}}) - \text{Eval}(\tau, \mathcal{F}) < \sum_{a \in \mathcal{F} - \hat{\mathcal{F}}} \pi_a$.

Let $\mathcal{A} = \mathcal{F} - \hat{\mathcal{F}}$. From the algorithm, we know that the set $\mathcal{F}$ satisfies $\text{Eval}(\tau, \mathcal{F}) = \text{MinEval}(\tau, \mathcal{C})$. Since $\hat{\mathcal{F}} \subseteq \mathcal{C} - \mathcal{A}$, the definition of MinEval implies that $\text{MinEval}(\tau, \mathcal{C} - \mathcal{A}) \leq \text{Eval}(\tau, \hat{\mathcal{F}})$. Thus, we have $\text{MinEval}(\tau, \mathcal{C} - \mathcal{A}) - \text{MinEval}(\tau, \mathcal{C}) \leq \text{Eval}(\tau, \hat{\mathcal{F}}) - \text{Eval}(\tau, \mathcal{F}) < \sum_{a \in \mathcal{A}} \pi_a$, which shows that the inequality output by the algorithm is violated.

Lemma 14 *If π violates some inequality in Equations (1), (2), and (3), then the separation algorithm run on input π returns an inequality.*

PROOF. If the violation is in Equations (1) or (3), the violated inequality will be output by the direct check in step 1. If some inequality is output by step 2, we are done. Otherwise, since steps 1 and 2 output no inequality, we know that $\pi_a = 0$ for all $a \in \mathcal{C} - \mathcal{F}$, where $\mathcal{F}$ is as computed in the algorithm.

Now, suppose the inequality from Equation (2) for set $\mathcal{A} \subseteq \mathcal{C}$ is violated. In other words, $\sum_{a \in \mathcal{A}} \pi_a > \text{MinEval}(\tau, \mathcal{C} - \mathcal{A}) - \text{MinEval}(\tau, \mathcal{C})$. Let $\mathcal{B}$ be such that $\text{Eval}(\tau, \mathcal{B}) = \text{MinEval}(\tau, \mathcal{C} - \mathcal{A})$.

Thus, $\sum_{a \in \mathcal{A}} \pi_a > \text{MinEval}(\tau, \mathcal{C} - \mathcal{A}) - \text{MinEval}(\tau, \mathcal{C}) = \text{Eval}(\tau, \mathcal{B}) - \text{Eval}(\tau, \mathcal{F})$.

Since $\pi_a = 0$ for all $a \in \mathcal{C} - \mathcal{F}$, we have $\text{Eval}(\tau, \mathcal{F}) + \sum_{a \in \mathcal{F} \cap \mathcal{A}} \pi_a > \text{Eval}(\tau, \mathcal{B})$.

Adding $\sum_{a \in \mathcal{F} - \mathcal{A}} \pi_a$ to both sides of the above inequality and substituting the definition of Eval, we have $\text{Bcost}(\mathcal{F}) + \sum_{a \in \mathcal{F}} \tau(a) + \sum_{a \in \mathcal{F}} \pi_a > \text{Bcost}(\mathcal{B}) + \sum_{a \in \mathcal{B}} \tau(a) + \sum_{a \in \mathcal{F} - \mathcal{A}} \pi_a$.

Since $\pi_a = 0$ for all $a \in \mathcal{C} - \mathcal{F}$ and $\mathcal{B} \subseteq \mathcal{C} - \mathcal{A}$, we can alter the right hand side of the above inequality to get $\text{Bcost}(\mathcal{F}) + \sum_{a \in \mathcal{F}} \tau(a) + \sum_{a \in \mathcal{F}} \pi_a > \text{Bcost}(\mathcal{B}) + \sum_{a \in \mathcal{B}} \tau(a) + \sum_{a \in \mathcal{B}} \pi_a + \sum_{a \in \mathcal{F} - \mathcal{A} - \mathcal{B}} \pi_a$.

By applying the definition of $\hat{\tau}$ and Eval, we have $\text{Eval}(\hat{\tau}, \mathcal{F}) > \text{Eval}(\hat{\tau}, \mathcal{B}) +$

$\sum_{a \in \mathcal{F}-\mathcal{A}-\mathcal{B}} \pi_a$. We know that $\pi_a \geq 0$ for all $a \in \mathcal{P}$ since the algorithm does not output anything in step 1. Thus, $\text{Eval}(\hat{\tau}, \mathcal{F}) > \text{Eval}(\hat{\tau}, \mathcal{B}) \geq \text{MinEval}(\hat{\tau}, \mathcal{C}) = \text{Eval}(\hat{\tau}, \hat{\mathcal{F}})$, where $\hat{\mathcal{F}}$ is as computed in the algorithm. So, step 3 outputs an inequality.

The following lemma illustrates a key difference between Garg et al. and this work.

Lemma 15 *If suppliers are substitutes, then all but the $|\mathcal{C}|$ singleton equations of Equation (2) are not constraining. Thus, if suppliers are substitutes, optimization over the core of the buyer-supplier game is reduced to solving a polynomially sized linear program.*

PROOF. Suppose that the suppliers are substitutes. By the definition of "suppliers are substitutes", $\tilde{V}(\mathcal{P}) - \tilde{V}(\mathcal{P} - \mathcal{A}) \geq \sum_{a \in \mathcal{A}} [\tilde{V}(\mathcal{P}) - \tilde{V}(\mathcal{P} - \{a\})]$ for all $\mathcal{A} \subseteq \mathcal{C}$. By the definition of $\tilde{V}$,

$$\begin{aligned}&\text{MinEval}(\tau, \text{Bcost}, \mathcal{C} - \mathcal{A}) - \text{MinEval}(\tau, \text{Bcost}, \mathcal{C}) \\ &\geq \sum_{a \in \mathcal{A}} [\text{MinEval}(\tau, \text{Bcost}, \mathcal{C} - \{a\}) - \text{MinEval}(\tau, \text{Bcost}, \mathcal{C})]\end{aligned}$$

for all $\mathcal{A} \subseteq \mathcal{C}$. This implies that if the singleton equations in Equation (2) are satisfied, then so are all equations in Equation (2). Thus, if suppliers are substitutes, we may drop all non-singleton equations from Equation (2) and reduce the number of inequalities to a polynomial in the number of players.

5 Inapproximability of Optimization Over Core Solutions

Consider a buyer-supplier game defined by $(\mathcal{C}, \tau, \text{Bcost})$. We introduced the concept of the focus point price in the introduction. The concept leads us to ask the natural question: What is the difference between the best and worst core outcome for the buyer? In other words, the value of interest is the solution to the linear program: maximize $\sum_{a \in \mathcal{C}} \pi_a$ subject to Equations (1), (2), and (3). This natural question leads us to define the focus point price (FPP) problem as follows: on input $(\mathcal{C}, \tau, \text{Bcost})$, output the optimal value of the afore mentioned linear program.

Define the Necessary Element (NEL) problem as follows. Given parameters $(\mathcal{C}, \tau, \text{Bcost})$ return TRUE if there exist an element $a \in \mathcal{C}$ such that for all $\mathcal{F} \subseteq \mathcal{C}$ satisfying $\text{Eval}(\tau, \text{Bcost}, \mathcal{F}) = \text{MinEval}(\tau, \text{Bcost}, \mathcal{C})$ we have $a \in \mathcal{F}$. Otherwise, return FALSE.

Define the OPT-SET problem as follows. Given parameters $(\mathcal{C}, \tau, \text{Bcost})$, return $\mathcal{F}$ such that $\text{Eval}(\tau, \text{Bcost}, \mathcal{F}) = \text{MinEval}(\tau, \text{Bcost}, \mathcal{C})$.

In this section, we will show that the FPP problem, the OPT-SET problem and the NEL problem are polynomial time equivalent. In Section 5.1, we show how to solve the OPT-SET problem in polynomial time if the NEL problem is solvable in polynomial time. In Section 5.2, we show the polynomial time equivalence of NEL, OPT-SET, and separation over Equations (1), (2), and (3).

5.1 Polynomial Time Reduction from OPT-SET to NEL

In this section, we show that given a polynomial time algorithm to solve the NEL problem, we can solve the OPT-SET problem in polynomial time. All of the lemmas in this section are used solely to prove the section's main result, Lemma 20.

For a fixed tuple $(\mathcal{C}, \tau, \text{Bcost})$ we say we extend the tuple to contain a *shadow element* for an element $a \subseteq \mathcal{C}$ by creating the extended tuple $(\hat{\mathcal{C}}, \hat{\tau}, \text{Bcost}^*)$, where $\hat{\mathcal{C}} = \mathcal{C} \cup b$ with $b \notin \mathcal{C}$; $\hat{\tau}$ is the same as τ with the addition that $\hat{\tau}(b) = \tau(a)$; and for $\mathcal{A} \subseteq \hat{\mathcal{C}}$, if $b \notin \mathcal{A}$, then $\text{Bcost}^*(\mathcal{A}) = \text{Bcost}(\mathcal{A})$, otherwise $\text{Bcost}^*(\mathcal{A}) = \text{Bcost}((\mathcal{A} - \{b\}) \cup \{a\})$. We call b the *shadow element* corresponding to a.

The *full shadow extension* of $(\mathcal{C}, \tau, \text{Bcost})$ is the tuple $(\hat{\mathcal{C}}, \hat{\tau}, \text{Bcost}^*)$ resulting from extending $(\mathcal{C}, \tau, \text{Bcost})$ to contain a shadow element for each element in $\mathcal{C}$.

First, we reduce OPT-SET to NEL. To show the result, we analyze the following algorithm, which we call the shadow algorithm.

On input $(\mathcal{C}, \tau, \text{Bcost})$,

```
1 Let $(\hat{\mathcal{C}}^*, \hat{\tau}, \text{Bcost}^*)$ be the full shadow extension of $(\mathcal{C}, \tau, \text{Bcost})$. Let the
  program variable $\hat{\mathcal{C}}$ equal $\hat{\mathcal{C}}^*$.
2 For each $a \in \mathcal{C}$
    • Remove $a$'s corresponding shadow element from $\hat{\mathcal{C}}$.
    • Run NEL on $(\hat{\mathcal{C}}, \hat{\tau}, \text{Bcost}^*)$.
    • If the return value is TRUE, then add the shadow element back to $\hat{\mathcal{C}}$.
    • If the return value is FALSE, then remove $a$ from $\hat{\mathcal{C}}$.
3 Return $\hat{\mathcal{C}} \cap \mathcal{C}$. In other words, we return all elements from $\mathcal{C}$ remaining in
  $\hat{\mathcal{C}}$, disregarding any shadow elements.
```

Lemma 16 *Let* $(\mathcal{C}, \tau, \text{Bcost})$ *be the input to the shadow algorithm. Let the* *triple* $(\hat{\mathcal{C}}^*, \hat{\tau}, \mathrm{Bcost}^*)$ *be the full shadow extension of* $(\mathcal{C}, \tau, \mathrm{Bcost})$*. If for all* $\mathcal{A} \subseteq \hat{\mathcal{C}}^*$ *the NEL problem on input* $(\mathcal{A}, \hat{\tau}, \mathrm{Bcost}^*)$ *is solvable in polynomial time, then the shadow algorithm runs in polynomial time.*

PROOF. Creating $\hat{\mathcal{C}}^*$ takes polynomial time since there are $O(|\mathcal{C}|)$ elements. Defining $\hat{\tau}$ takes polynomial time since there are $O(|\mathcal{C}|)$ inputs. Queries to Bcost^* can be implemented with polynomial overhead on top of queries to Bcost. Thus, the initialization step of the algorithm takes polynomial time.

Consider a single loop iteration. The first, third and forth lines of the loop each take $O(|\mathcal{C}|)$ time. The second step takes polynomial time by the lemma assumption. Thus, a single loop iteration takes polynomial time.

There are $|\mathcal{C}|$ loop iterations and computing the intersection in the algorithm's final step takes $O(|\mathcal{C}|)$ time. Thus the algorithm runs in polynomial time.

Lemma 17 *The shadow algorithm maintains the invariant*

$$\mathrm{MinEval}(\tau, \mathrm{Bcost}, \mathcal{C}) = \mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}}).$$

PROOF. Initially, $\mathrm{MinEval}(\tau, \mathrm{Bcost}, \mathcal{C}) = \mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}})$ by the definitions of $\hat{\mathcal{C}}$, $\hat{\tau}$, and Bcost^*.

Consider the loop iteration for $a \in \mathcal{C}$. Let the corresponding shadow element be b. When we remove or add b to $\hat{\mathcal{C}}$, we have $\mathrm{MinEval}(\tau, \mathrm{Bcost}, \mathcal{C}) = \mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}})$ by the definitions of $\hat{\tau}$, and Bcost^* and the fact that a is still in $\hat{\mathcal{C}}$.

We only remove both a and b if NEL returned FALSE before the removal of a. Let $\hat{\mathcal{C}}_a$ and $\hat{\mathcal{C}}'_a$ be the value of the variable $\hat{\mathcal{C}}$ before and after the removal of a, respectively. Since NEL returned FALSE on $(\hat{\mathcal{C}}_a, \hat{\tau}, \mathrm{Bcost}^*)$, there exists some $\mathcal{F} \subseteq \hat{\mathcal{C}}_a$ such that $a \notin \mathcal{F}$ and $\mathrm{Eval}(\hat{\tau}, \mathrm{Bcost}^*, \mathcal{F}) = \mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}}_a)$. Thus, $\mathcal{F} \subseteq \hat{\mathcal{C}}'_a$ and $\mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}}_a) = \mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}}'_a)$.

Thus, throughout the algorithm the value of $\mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}})$ does not change, which concludes the proof.

Lemma 18 *Let* $\hat{\mathcal{C}}_a$ *be the value of the variable* $\hat{\mathcal{C}}$ *at the end of iteration corresponding to* $a \in \mathcal{C}$*. If* $a \in \hat{\mathcal{C}}_a$*, then* a *is in all OPT-SET solutions on input* $(\mathcal{A}, \hat{\tau}, \mathrm{Bcost}^*)$ *where* $\mathcal{A} = \hat{\mathcal{C}}_a \cap \mathcal{C}$*.*

PROOF. Let the arguments of the NEL problem which is solved during the iteration corresponding to a be $(\mathcal{B}, \hat{\tau}, \mathrm{Bcost}^*)$.

Since $a \in \hat{\mathcal{C}}_a$, NEL returns TRUE during the iteration corresponding to a.

Suppose there is a solution $\mathcal{F} \subseteq \mathcal{C}$ to OPT-SET on input $(\mathcal{A}, \hat{\tau}, \mathrm{Bcost}^*)$ which does not contain a. Consider $\mathcal{F}$ and $\hat{\mathcal{F}}$ where $\hat{\mathcal{F}}$ contains all the shadow elements of the elements of $\mathcal{F}$. The sets $\mathcal{F}$ and $\hat{\mathcal{F}}$ are disjoint and both subsets of $\mathcal{B}$. Also, by the definition of $\mathcal{F}$, $\hat{\tau}$ and Bcost^*, $\mathrm{Eval}(\hat{\tau}, \mathrm{Bcost}^*, \mathcal{F}) = \mathrm{Eval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{F}}) = \mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \mathcal{B})$. Thus, the NEL problem run during the iteration corresponding to a should return FALSE, which is a contradiction.

Lemma 19 *Let $\hat{\mathcal{C}}_a$ be the value of the variable $\hat{\mathcal{C}}$ at the end of iteration corresponding to $a \in \mathcal{C}$. We have* $\mathrm{MinEval}(\tau, \mathrm{Bcost}, \mathcal{C}) = \mathrm{MinEval}(\tau, \mathrm{Bcost}, \hat{\mathcal{C}}_a \cap \mathcal{C})$.

PROOF. Let $\mathcal{F}$ be such that $\mathrm{Eval}(\hat{\tau}, \mathrm{Bcost}^*, \mathcal{F}) = \mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}}_a)$. If $\mathrm{Eval}(\hat{\tau}, \mathrm{Bcost}^*, \mathcal{F}) = M$, then let $\hat{\mathcal{F}} = \emptyset$; otherwise, let $\hat{\mathcal{F}}$ be $\mathcal{F}$ with each shadow element replaced by the corresponding element in $\mathcal{C}$. By the definitions of $\hat{\tau}$ and Bcost^*, we have $\mathrm{Eval}(\hat{\tau}, \mathrm{Bcost}^*, \mathcal{F}) = \mathrm{Eval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{F}})$. But, by the construction of $\hat{\mathcal{F}}$, we have $\hat{\mathcal{F}} \subseteq \hat{\mathcal{C}}_a \cap \mathcal{C}$.

Thus,

$$\begin{aligned}\mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}}_a) &= \mathrm{Eval}(\hat{\tau}, \mathrm{Bcost}^*, \mathcal{F}) \\ &= \mathrm{Eval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{F}}) \quad \geq \mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}}_a \cap \mathcal{C}).\end{aligned}$$

Also, by the definition of MinEval, we have that $\mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}}_a)$ is at most $\mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}}_a \cap \mathcal{C})$. So,

$$\mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}}_a) = \mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}}_a \cap \mathcal{C}).$$

Combining Lemma 17 with the result from the last paragraph,

$$\mathrm{MinEval}(\tau, \mathrm{Bcost}, \mathcal{C}) = \mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}}_a) = \mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}}_a \cap \mathcal{C}).$$

And, by the definition of $\hat{\tau}$ and Bcost^*, we have $\mathrm{MinEval}(\hat{\tau}, \mathrm{Bcost}^*, \hat{\mathcal{C}}_a \cap \mathcal{C})$ equals $\mathrm{MinEval}(\tau, \mathrm{Bcost}, \hat{\mathcal{C}}_a \cap \mathcal{C})$.

Lemma 20 *Let* $(\mathcal{C}, \tau, \mathrm{Bcost})$ *be the input to the shadow algorithm. Let the triple* $(\hat{\mathcal{C}}^*, \hat{\tau}, \mathrm{Bcost}^*)$ *be the full shadow extension of* $(\mathcal{C}, \tau, \mathrm{Bcost})$. *If for all $\mathcal{A} \subseteq \hat{\mathcal{C}}^*$ the NEL problem on input* $(\mathcal{A}, \hat{\tau}, \mathrm{Bcost}^*)$ *is solvable in polynomial time, then the OPT-SET problem on input* $(\mathcal{C}, \tau, \mathrm{Bcost})$ *is solvable in polynomial time.*

PROOF. By Lemma 16, the shadow algorithm runs in polynomial time.

Let $\hat{\mathcal{C}}'$ be the value of the variable $\hat{\mathcal{C}}$ at the end of the algorithm. By Lemma 19, $\hat{\mathcal{C}}' \cap \mathcal{C}$ is a superset of a solution to the OPT-SET problem. By Lemma 18, $\hat{\mathcal{C}}' \cap \mathcal{C}$ is a subset of a solution to the OPT-SET problem. Thus, the value returned by the shadow algorithm, $\hat{\mathcal{C}}' \cap \mathcal{C}$, is a solution to the OPT-SET problem.

5.2 Polynomial Time Equivalence of NEL, OPT-SET, and Separation

In this section we show a polynomial time equivalence between the NEL problem, the OPT-SET problem and the separation problem over Equations (1), (2), and (3). The lemmas in this section are used to show the section's main result, Theorem 23. As a byproduct of the proofs, we also show an hardness of approximation result for the FPP problem with Lemma 24.

Lemma 21 *The solution to the FPP problem on input* $(\mathcal{C}, \tau, \mathrm{Bcost})$ *is 0 if and only if the solution to the NEL problem on input* $(\mathcal{C}, \tau, \mathrm{Bcost})$ *is FALSE.*

PROOF. First, we prove that if the solution to the NEL problem is FALSE, then the solution to the FPP problem is zero. Consider all of the inequality pairs $\pi_a \leq \mathrm{MinEval}(\tau, \mathrm{Bcost}, \mathcal{C} - \{a\}) - \mathrm{MinEval}(\tau, \mathrm{Bcost}, \mathcal{C})$ and $\pi_a \geq 0$. Since the solution to the NEL problem is FALSE, for each $a \in \mathcal{C}$ there is a $\mathcal{F}_a \subseteq \mathcal{C}$ such that $\mathrm{Eval}(\mathcal{F}_a, \mathrm{Bcost}, \mathcal{C}) = \mathrm{MinEval}(\tau, \mathrm{Bcost}, \mathcal{C})$ and $a \notin \mathcal{F}_a$. Thus the first inequality in the pair reduces to $\pi_a \leq 0$, and the pair of inequalities imply $\pi_a = 0$. This is true for all $a \in \mathcal{C}$. Thus, the optimal value of the FPP linear program is zero.

Second, we prove that if the solution to the NEL problem is TRUE, then the solution to the FPP problem is greater than zero. If the solution to NEL is TRUE, then there is some $a \in \mathcal{C}$ such that if $\mathrm{Eval}(\tau, \mathrm{Bcost}, \mathcal{F}) = \mathrm{MinEval}(\tau, \mathrm{Bcost}, \mathcal{C})$ then $a \in \mathcal{F}$. In other words, a is in all solutions to the OPT-SET problem on input $(\mathcal{C}, \tau, \mathrm{Bcost})$.

Thus, for all $\mathcal{A} \subseteq \mathcal{C}$ with $a \in \mathcal{A}$,

$$\mathrm{MinEval}(\tau, \mathrm{Bcost}, \mathcal{C} - \mathcal{A}) - \mathrm{MinEval}(\tau, \mathrm{Bcost}, \mathcal{C}) > 0.$$

Let $\lambda = \min_{\substack{\mathcal{A} \subseteq \mathcal{C} \\ a \in \mathcal{A}}} [\mathrm{MinEval}(\tau, \mathrm{Bcost}, \mathcal{C} - \mathcal{A}) - \mathrm{MinEval}(\tau, \mathrm{Bcost}, \mathcal{C})]$. Consider the vector π with $\pi_b = 0$ for all $b \in \mathcal{C} - \{a\}$ and $\pi_a = \lambda$ and $\pi_\mu = M - \mathrm{MinEval}(\tau, \mathrm{Bcost}, \mathcal{C}) - \lambda$. Since $\lambda \leq M - \mathrm{MinEval}(\tau, \mathrm{Bcost}, \mathcal{C})$, this vector is feasible in the focus point price linear program and achieves an objective function value greater than zero.

Lemma 22 *If it is possible to approximate the solution to the FPP problem on input* $(\mathcal{C}, \tau, \mathrm{Bcost})$ *within any multiplicative factor, then the NEL problem on input* $(\mathcal{C}, \tau, \mathrm{Bcost})$ *is solvable in polynomial time.*

PROOF. Follows from Lemma 21.

A set of $(\mathcal{C}, \tau, \mathrm{Bcost})$ instances is *proper* if the following conditions hold:

- Given that $(\mathcal{C}, \tau, \mathrm{Bcost})$ is in the set, then so is $(\mathcal{C}, \hat{\tau}, \mathrm{Bcost})$, where $\hat{\tau}(a) = \tau(a) + \pi_a$ for a vector $\pi \in \Re_+^{|\mathcal{C}|}$.
- Given that $(\mathcal{C}, \tau, \mathrm{Bcost})$ is in the set, then so is $(\mathcal{A}, \hat{\tau}, \mathrm{Bcost}^*)$, where the triple $(\hat{\mathcal{C}}, \hat{\tau}, \mathrm{Bcost}^*)$ is the full shadow extension of $(\mathcal{C}, \tau, \mathrm{Bcost})$ and $\mathcal{A}$ is a subset of $\hat{\mathcal{C}}$.

The definition of proper has a natural interpretation when applied to the transformations of combinatorial minimization problems to buyer-supplier games. For example, for the shortest path problem, the first condition implies that the set of instances is closed with respect to lengthening the edges of the graph. On the other hand, the second condition implies that the set of instances is closed with respect to adding parallel edges or removing a subset of the edges.

Theorem 23 *On a proper set of instances, the separation problem over Equations (1), (2), and (3), the NEL problem and the OPT-SET problem are polynomial time equivalent.*

PROOF. If we can solve the NEL problem on a proper set of instances in polynomial time, then, by Lemma 20, we can solve the OPT-SET problem in polynomial time.

If we can solve the OPT-SET problem on a proper set of instances in polynomial time, then, by Theorem 12, we can solve the separation problem over Equations (1), (2), and (3) in polynomial time.

If we can solve the separation problem over Equations (1), (2), and (3) on a proper set of instances in polynomial time, then, by the polynomial time equivalence of separation and optimization, we can optimize linear objective functions over Equations (1), (2), and (3) in polynomial time. If we can optimize linear objective functions in polynomial time, by Lemma 22 we can solve the NEL problem in polynomial time.

Lemma 24 *On a proper set of instances, if it is not possible to solve the OPT-SET problem in polynomial time, it is not possible to approximate the* *solution to the FPP problem to within any multiplicative factor in polynomial time.*

PROOF. Follows from Theorem 23 and Lemma 22.

6 A Complementary Combinatorial Algorithm

In this section, we present an efficient combinatorial algorithm for solving the FPP problem for the buyer-supplier minimum spanning tree (MST) game. Before we go on to give the combinatorial algorithm, we present a useful simplification of the linear program representing the FPP problem for general buyer-supplier games that may be of independent interest.

For the simplification, fix a buyer-supplier game defined by $(\mathcal{C}, \tau, \mathrm{Bcost})$. Let the buyer-supplier game be derived from the combinatorial minimization problem MinProb as described in Section 1. We omit the parameters τ and Bcost from MinEval since they are fixed by the game.

Call the linear program the linear program from the FPP problem LP1 and let its optimal value be O_1. Consider the following linear program

$$\begin{aligned} \max \sum_{b \in \mathcal{C}} \pi_b & & \\ \text{s.t.} \sum_{b \in \mathcal{A}} \pi_b &\leq \mathrm{MinProb}(\mathcal{C} - \mathcal{A}) - \mathrm{MinProb}(\mathcal{C}) \qquad && \text{for all } \mathcal{A} \subseteq \mathcal{C} \\ \pi_b &\geq 0 && \text{for all } b \in \mathcal{C}. \end{aligned}$$

Call the above linear program LP2 and let its optimal value be O_2. Using the definitions of MinProb, MinEval, and simple manipulations of the linear programs, it is straight forward to show the following three lemmas.

Lemma 25 *If* $\mathrm{MinProb}(\mathcal{C}) \geq M$*, then* $O_1 = 0$.

Lemma 26 *If* $\mathrm{MinProb}(\mathcal{C}) < M$ *and* $O_2 \leq M - \mathrm{MinProb}(\mathcal{C})$*, then* $O_1 = O_2$.

Lemma 27 *If* $\mathrm{MinProb}(\mathcal{C}) < M$ *and* $O_2 > M - \mathrm{MinProb}(\mathcal{C})$*, then* $O_1 = M - \mathrm{MinProb}(\mathcal{C})$.

By the three preceding lemmas, finding the value of $\mathrm{MinProb}(\mathcal{C})$ and the solution to LP2 is sufficient to solve the FPP problem for a given buyer-supplier game.

We continue with the combinatorial algorithm. Let a graph $G = (\mathcal{V}, \mathcal{E})$ and edge weights $w : \mathcal{E} \rightarrow \Re_+$ be given. Let $\mathrm{MSTVal} : 2^{\mathcal{E}} \rightarrow \Re_+$ be a function that takes as input a set of the edges $\mathcal{A} \subseteq \mathcal{E}$ and returns the weight of the minimum spanning tree of the graph induced by the edges of $\mathcal{A}$. If no spanning tree exists, MSTVal returns ∞.

By the transformation in Section 1 and Lemma 11 in the buyer-supplier minimum spanning tree game, $\mathcal{C} = \mathcal{E}$, $\tau(a) = w(a)$, and $\text{Bcost}(\mathcal{A}) = M$ if $\mathcal{A}$ does not connect all nodes in $\mathcal{V}$, or 0 otherwise. We omit the parameters τ and Bcost from MinEval, since they are fixed by the game.

We begin with some basic results on MSTs, introduced here without proof as they can be found in or derived from principles found in many graduate texts such as Cormen et al. [5].

Lemma 28 *Let $H = (\mathcal{V}_1, \mathcal{E}_1)$ be any graph. Let $T = (\mathcal{V}_1, \mathcal{E}_1')$ be a minimum spanning tree of the graph H. For $e \in \mathcal{E}_1'$ let the cut created in T by the removal of e be $(\mathcal{A}_e, \mathcal{B}_e)$ for some $\mathcal{A}_e \subseteq \mathcal{V}_1$ and $\mathcal{B}_e \subseteq \mathcal{V}_1$.*

- *The edge e is a minimum weight edge spanning the cut $(\mathcal{A}_e, \mathcal{B}_e)$.*
- *The tree T restricted to vertices in $\mathcal{A}_e$ is a minimum spanning tree of the graph induced by the vertices $\mathcal{A}_e$. A symmetric statement is true for $\mathcal{B}_e$*
- *Let $\mathcal{U}$ be the set of edges spanning the cut and let $a = arg\ min_{b \in \mathcal{U} - \{e\}} w(b)$. We have $w(a) - w(e) = \text{MSTVal}(\mathcal{E}_1 - \{e\}) - \text{MSTVal}(\mathcal{E}_1)$.*

The following lemma is due to Bikhchandani et al., who show that in the MST buyer-supplier game, the suppliers are substitutes [2].

Lemma 29 *Suppose the graph G is connected and thus $\text{MSTVal}(\mathcal{E})$ is finite. For all $\mathcal{A} \subseteq \mathcal{E}$, $\text{MSTVal}(\mathcal{E} - \mathcal{A}) - \text{MSTVal}(\mathcal{E}) \geq \sum_{e \in \mathcal{A}}[\text{MSTVal}(\mathcal{E} - \{e\}) - \text{MSTVal}(\mathcal{E})]$*

Let LP2 be as defined above with MinProb equal to MSTVal.

Lemma 30 *The inequalities corresponding to singleton sets $\{e\}$ for $e \in \mathcal{E}$ form an optimal basis for* LP2. *In particular, setting $\pi_e = \text{MSTVal}(\mathcal{E} - \{e\}) - \text{MSTVal}(\mathcal{E})$ for all $e \in \mathcal{E}$ gives an optimal vector for* LP2.

PROOF. For $\mathcal{A} \subseteq \mathcal{E}$, by Lemma 29, we have

$$\begin{aligned}\sum_{e \in \mathcal{A}} \pi_e &= \sum_{e \in \mathcal{A}} [\text{MSTVal}(\mathcal{E} - \{e\}) - \text{MSTVal}(\mathcal{E})] \\ &\leq \text{MSTVal}(\mathcal{E} - \mathcal{A}) - \text{MSTVal}(\mathcal{E}).\end{aligned}$$

Thus, π is feasible in LP2. Each inequality of type $\pi_e \leq \text{MSTVal}(\mathcal{E} - \{e\}) - \text{MSTVal}(\mathcal{E})$ is tight, thus it is not possible to increase any coordinate of π. Thus π is an optimal vector and the singleton inequalities form an optimal basis.

We give a modified Kruskal Algorithm which can be used to compute the optimal value of LP2.

Run Kruskal Algorithm with the following modifications.

- Throughout the algorithm's execution we will keep an auxiliary set of edges, $\mathcal{A}$, which is initially empty.
- When edge e is added to the minimum spanning forest, also add e to the set $\mathcal{A}$.
- Suppose edge e is rejected from addition to the minimum spanning forest because it creates a cycle. Let the cycle created be $H = (\mathcal{V}', \mathcal{E}')$. For each edge $a \in \mathcal{E}' - \{e\}$, if $a \in \mathcal{A}$, label a with $w(e) - w(a)$ and remove a from $\mathcal{A}$.

Lemma 31 *Let G be connected and $T = (\mathcal{V}_1, \mathcal{E}_1)$ be the minimum spanning tree computed by the modified Kruskal Algorithm when it is run on $G = (\mathcal{V}, \mathcal{E})$. If $e \in \mathcal{E}_1$ has been labeled, the label is equal to* $\mathrm{MSTVal}(\mathcal{E} - \{e\}) - \mathrm{MSTVal}(\mathcal{E})$. *Otherwise,* $\mathrm{MSTVal}(\mathcal{E} - \{e\}) - \mathrm{MSTVal}(\mathcal{E}) = \infty$.

PROOF. Let $\mathcal{U}$ be the set of edges spanning the cut created in T by the removal of e. Also, let a' be any arg $\min_{b \in \mathcal{U}-\{e\}} w(b)$. By Lemma 28, $w(a') - w(e) = \mathrm{MSTVal}(\mathcal{E} - \{e\}) - \mathrm{MSTVal}(\mathcal{E})$.

Let e be labeled by the modified Kruskal Algorithm when the edge a creates a cycle. For the first part of the lemma, all we must show is $a = a'$.

Let $\mathcal{K}$ be the set of edges which create a cycle involving e during the algorithm. If $b \in \mathcal{E} - \mathcal{U}$, then b does not span the cut created in T by the removal of e. Thus, both vertices of b lie on the same side of the cut. Thus, b cannot create a cycle involving e, since all edges in the cycle except b must be in the tree T. Also, the edge e cannot form a cycle with itself. Thus, $\mathcal{K} \subseteq \mathcal{U} - \{e\}$.

Consider any $b \in \mathcal{U} - \{e\}$. The edge b must be rejected by the modified Kruskal algorithm, otherwise the edge would be in T and wouldn't span the cut created in T by the removal of e. Thus, b must create a cycle during the algorithm. Since all edges except b which make up the cycle must be in T and b spans the cut created by e, the cycle created by b includes e. Thus, $\mathcal{U} - \{e\} \subseteq \mathcal{K}$, and $\mathcal{U} - \{e\} = \mathcal{K}$.

Since the modified Kruskal Algorithm process edges of G in ascending order of weight and e is labeled by the element from $\mathcal{K}$ which is processed first, we know $a =$ arg $\min_{b \in \mathcal{K}} w(b)$. Since $\mathcal{K} = \mathcal{U}$, we also have $a = a'$ and we have shown the first part of the lemma.

If e is not labeled, then $\mathcal{K}$ must be empty. Since $\mathcal{K} = \mathcal{U}$, the set $\mathcal{U}$ must also be empty. And thus, the removal of e must disconnect the graph G. Thus,

$$\text{MSTVal}(\mathcal{E} - \{e\}) - \text{MSTVal}(\mathcal{E}) = \infty.$$

The modified Kruskal algorithm produces the optimal value to LP2. By our simplification of the FPP linear program, the MST weight of G and the optimal value of LP2 solve the FPP problem for the buyer-supplier MST game.

References

[1] A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 482–491, Oct. 2001.

[2] S. Bikhchandani, S. de Vries, J. Schummer, and R. Vohra. Linear programming and vickrey auctions. In *Mathematics of the Internet: E-Auction and Markets*, pages 75–115. Springer-Verlag, New York, NY, 2002.

[3] S. Bikhchandani and J. Ostroy. The package assignment model. *Journal of Economic Theory*, 107:377–406, 2002.

[4] A. Caprara and A. N. Letchford. Computing good allocations for combinatorial optimization games. Lancaster University Management School working paper. Available from http://eprints.lancs.ac.uk/7082/, 2006.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2001.

[6] X. Deng, T. Ibaraki, and H. Nagamochi. Algorithmic aspects of the core of combinatorial optimization games. *Mathematics of Operations Research*, 24:751–766, 1999.

[7] X. Deng, C. Papadimitriou, and Safra S. On the complexity of equilibria. In *Proceedings of the 34th annual ACM symposium on Theory of Computing*, pages 67–71, Montreal, Quebec, 2002.

[8] F. Y. Edgeworth. *Mathematical psychics, an essay on the application of mathematics to the moral sciences*. A. M. Kelley, New York, NY, 1961.

[9] U. Faigle and W. Kern. On the core of ordered submodular cost games. *Mathematical Programming*, 87:483–499, 2000.

[10] R. Garg, V. Kumar, A. Rudra, and A. Verma. Coalitional games on graphs: core structure, substitutes and frugality. In *Proceedings of the 4th ACM conference on Electronic Commerce*, pages 248–249, San Diego, CA, 2003.

[11] D. B. Gillies. *Some Theorems on n-Person Games*. PhD thesis, Princeton University, 1953.

[12] M. X. Goemans and M. Skutella. Cooperative facility location games. *Journal of Algorithms*, 50:194–214, 2004.

[13] A. R. Karlin, D. Kempe, and T. Tamir. Beyond VCG: frugality of truthful mechanisms. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 615–624, Oct. 2005.

[14] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley and Sons, New York, NY, 1988.

[15] M. Pál and E. Tardos. Group strategy proof mechanisms via primal-dual algorithms. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 584–593, Oct. 2003.

[16] C. H. Papadimitriou. Algorithms, games, and the internet. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 749–753, July 2001.

[17] C. H. Papadimitriou and T. Roughgarden. Computing equilibria in multi-player games. In *Proceedings of the 16th Annual ACM-SIAM symposium on Discrete algorithms*, pages 82–91, January 2005.

[18] T. Roughgarden. *Selfish Routing and the Price of Anarchy.* MIT Press, Cambridge, MA, 2005.

[19] T. Roughgarden and E. Tardos. How bad is selfish routing? *J. ACM*, 49:236–259, 2002.

[20] T. Roughgarden and E. Tardos. Bounding the inefficiency of equilibria in nonatomic congestion games. *Games and Economic Behaviour*, 47:389–403, 2004.

[21] L. S. Shapley. Notes on the n-person game III: Some variants of the von Neumann-Morgenstern definition of solution. Research memorandum, RAND Corporation, Santa Monica, CA, 1952.

[22] L. S. Shapley and M. Shubik. The assignment game i: The core. *International Journal of Game Theory*, 1:111–130, 1972.

[23] M. Shubik. *Game Theory In The Social Sciences.* MIT Press, Cambridge, Massachussetts, 1984.

[24] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, Princeton, NJ, 1953.