The Notion of a Rational Convex Program, and
an Algorithm for the Arrow-Debreu Nash Bargaining Game

*Vijay V. Vazirani**

Abstract

We introduce the notion of a *rational convex program (RCP)* and we classify the known RCPs into two classes: quadratic and logarithmic. The importance of rationality is that it opens up the possibility of computing an optimal solution to the program via an algorithm that is either combinatorial or uses an LP-oracle. Next we define a new Nash bargaining game, called **ADNB**, which is derived from the linear case of the Arrow-Debreu market model. We show that the convex program for **ADNB** is a logarithmic RCP, but unlike other known members of this class, it is non-total.

Our main result is a combinatorial, polynomial time algorithm for **ADNB**. It turns out that the reason for infeasibility of logarithmic RCPs is quite different from that for LPs and quadratic RCPs. We believe that our ideas for surmounting the new difficulties will be useful for dealing with other non-total RCPs as well. We give an application of our combinatorial algorithm for ADNB to an important "fair" throughput allocation problem on a wireless channel. Finally, we present a number of interesting questions that the new notion of RCP raises.

1   Introduction

Nash bargaining [Nas50] is a central solution concept within game theory for "fair" allocation of utility among competing players in the presence of complete information; it has numerous applications and a large following, e.g., see [Kal85, TL89, OR94]. In this paper, we define a very general Nash bargaining game, which is derived from the linear case of the Arrow-Debreu market model. The setup is the same as this model, but instead of resorting to the solution concept of a market equilibrium for reallocating goods among the agents, we resort to the Nash bargaining solution. We call this game the *Arrow-Debreu Nash Bargaining Game*, abbreviated **ADNB**. Our main result is a combinatorial[1], polynomial time algorithm for **ADNB**.

The solution to a Nash bargaining game is obtained by maximizing a concave function over a convex set, i.e., it is the solution to a convex program. It turns out that the convex program for **ADNB** has special structural properties that makes possible such an algorithm. Below we explain these properties in a larger context so as to point out the novelty and significance of deriving such an algorithm for **ADNB**.

1.1   Rational convex programs

The central problems of the field of combinatorial optimization, such as matching, flow and minimum spanning tree, share the feature that they possess LP-relaxations that always have integral optimal solutions. Let us call such an LP an *integral linear program* or ILP. In a sense, the solution produced by an ILP is qualitatively "better" than that produced by an arbitrary LP; in fact, it is similar in quality to the solution produced by an linear integer program. When LP solvers were not fast enough, this feature was not directly useful, and it made sense to seek efficient combinatorial algorithms for these problems. Now that LP-solvers have improved considerably, are combinatorial algorithms relevant anymore? To answer this question, one only needs to consider the highly acclaimed 3 volume series, *Combinatorial Optimization*, published by Schrijver a few years ago [Sch03].

The reason of course is that the design of these special purpose algorithms, for individual problems, have revealed a very rich theory that underlies not only combinatorics but also the theory of algorithms. Indeed, as far as the latter field is concerned, some of its formative and most fundamental notions, such as polynomial time solvability [Edm65], were conceived within combinatorial optimization. Additionally, over the years, ideas and notions developed within combinatorial optimization have helped spawn off new algorithmic areas such as approximation algorithms and parallel algorithms.

In the first part of this paper, we introduce the notion of a *rational convex program* or RCP – a nonlinear convex program that always has a rational optimal solution (see Section 2 for a formal definition). Again, the solution produced by an RCP is qualitatively "better" than that produced by an arbitrary convex program, and is like that produced by an LP. A convex program solver, based on either the ellipsoid algorithm or interior point methods [GLS88], can find an optimal solution to such a program in polynomial time; however, at present,

---

[1]In this paper, we will use "combinatorial" in the same sense as in [Sch03] (Volume A, page 1); such algorithms do not assume access to an LP or convex program solver.

convex program solvers are considerably slower than LP-solvers and are practical only for small instance sizes, e.g., see the benchmarks in [Mit]. Besides issues of efficiency, we claim that it will be a mistake if we do not explore RCPs further – indeed, we believe that in many ways, the situation is similar to that of ILPs. In both cases, the existence of much higher quality solution is indicative of combinatorial structure that can not only lead to efficient algorithms but also deep insights that yield unexpected gains. We are proposing the following two programs of study:

Program A: Design polynomial time (or better, strongly polynomial) combinatorial algorithms that solve individual RCPs.

Program B: Design polynomial time (or better, strongly polynomial) algorithms that solve individual RCPs, given an LP-oracle (each call counts as 1 step). Since LP-solvers are much faster than convex program solvers, such algorithms may be valuable in practice.

In Section 2 we formally define two classes of RCPs that have been studied so far. Convex programs belonging to the first class, quadratic RCPs, are obviously rational and have been studied for several decades. Although the first program belonging to the second class, logarithmic RCPs, was discovered in 1956 [EG59], the rest were found only in the last decade in the context of the fascinating question of understanding the computability of equilibria for various market models. Rationality for programs in this class is not automatic and has to be established piecemeal. Numerous open questions remain, not the least of which is to determine if there are other classes of RCPs.

We show that the program corresponding to **ADNB** is a rational convex program that lies in the second class. For our purposes, the novelty of this program lies in that whereas the previously known candidates of this class were all total, i.e., had finite optimal solutions for each setting of the parameters, this program is non-total. It turns out that the reason for infeasibility of logarithmic RCPs is quite different from that for LPs and quadratic RCPs. In the settings of ILPs and quadratic RCPs, the reason for infeasibility is that the polytope defined by the constraints is empty, and it turns out that designing a combinatorial algorithm for a non-total problem has been found to be no harder than that for a total problem[2]. However, a logarithmic RCP can be infeasible even though its polytope is guaranteed to be non-empty – the reason for infeasibility is that at each point in the polytope, the objective function is undefined. This is the case for **ADNB**.

1.2 A combinatorial algorithm for ADNB

In the second part of this paper, we give a polynomial time combinatorial algorithm for **ADNB**. Consider the set of instances of **ADNB** in which the disagreement utility of each agent is zero (see Section 3 for definitions and explanation). The convex programs corresponding to this set of instances are precisely those arising for linear Fisher markets in which each buyer has unit money. It is therefore natural that our algorithm builds on the combinatorial algorithm for linear Fisher markets [DPSV08]. The latter algorithm, which is based on the primal-dual

---

[2]As an example, recall algorithms for finding a maximum weight perfect matching (a non-total problem) and a maximum weight matching (a total problem) in a bipartite graph.

paradigm, starts with low prices that are guaranteed to be weakly dominated by equilibrium prices on each component. Selling all goods at these prices leaves buyers with surplus money, in general. The algorithm increases prices iteratively, in a principled manner, thereby decreasing the surplus money of buyers. It terminates when the surplus is driven down to zero. The notion of a balanced flow is used to establish polynomial running time.

Using KKT conditions of the convex program of **ADNB** and interpreting dual variables as prices, we can reduce a given instance it to a new, natural market model, which we call *flexible budget market*. This model differs from the linear Fisher market in the following ways: whereas the latter market always has an equilibrium, the former doesn't, and whereas in the latter market, each buyer has a fixed amount of money, in the former, the money of each buyer is a function (increasing) of the prices of goods. We note that in both these market models, equilibrium prices are unique. Now, it suffices to check if the reduced market instance has an equilibrium, i.e., is feasible, and if so find an equilibrium. This leads to the first new issue to be dealt with:

First difficulty: Determine if the given instance is feasible.

Our algorithm consists of two stages. Stage I starts with prices that are weakly dominated by equilibrium prices and it terminates with prices that help establish feasibility or infeasibility of the given instance; if feasible, Stage II computes equilibrium prices. The second stage is similar to the DPSV algorithm: it also systematically raises prices until the surplus is driven down to zero and it also uses balanced flow. The new difficulty in implementing this idea comes from the following fact. In a flexible budget market, the money of each buyer is an increasing function of the prices of goods. Hence, as the algorithm raises prices of goods, the money of buyers also increases – as a result, the surplus money of a buyer is not guaranteed to decrease!

Second difficulty: Ensure polynomial time termination of Stage II, even though the algorithm is "chasing a moving target".

1.3 Algorithmic contributions

We now give a high level description of ideas needed to deal with the two difficulties. For the first one, we give an LP that determines feasibility and we use its dual to prove, in the Main Lemma, Lemma 9, that non-zero prices such that the total surplus of all buyers is at least $n - \epsilon$, for a suitable ϵ, yield a proof that the given instance is infeasible; here n is the total number of buyers. On the other hand, prices such that the surplus of each buyer is less than 1 yield a proof of feasibility. In order to ensure polynomial time termination of Stage I, it is essential to ensure that ϵ is at least inverse exponential – our proof of rationality of the convex program for **ADNB** yields this bound.

Now, the ostensible goal of Stage I is to arrive at a proof of infeasibility. This requires increasing the total surplus of buyers and this can be achieved by systematically decreasing prices. However, if the given instance is feasible, as Stage I proceeds, the low surplus buyers will manage to "neutralize" the surplus of high surplus buyers and eventually drive down the surplus of each buyer to less than 1, hence giving a proof of feasibility. The exact mechanism is quite intricate and making it work in polynomial time is challenging – it requires using balanced flow in a different manner than in [DPSV08] or in our Stage II. In particular, the potential

4

function used for showing polynomial time termination of Stage I does not have a fixed number of terms (see Section 11.1). Observe that unlike Stage II, which monotonically increases prices, Stage I decreases prices, even though current prices are dominated by equilibrium prices (if the instance is feasible). We believe that some of these ideas will be useful for dealing with other non-total RCPs as well.

For dealing with the second difficulty, we prove that if Stage II is started with prices satisfying the feasibility condition stated above, then as prices are increased, the money of buyers increases slowly enough that the surplus keeps decreasing. We then give a potential function argument to show that the surplus must drop to zero in polynomial time.

An obvious question is the following. Is Stage I really needed? Why not simply run Stage II and if it fails to give a solution, we would realize that the instance is infeasible. It turns out that if Stage II is run with arbitrary initial prices, it is not guaranteed to terminate, even if the given instance is feasible. The reason lies in the second difficulty – when started with arbitrary initial prices, as prices are increased, the money of buyers may increase even faster, thereby increasing the surplus instead of decreasing it. Thus Stage I is essential even for solving an instance that is guaranteed to be feasible, i.e., the promise problem[3].

We note that Tseng and Bertsekas [TB00] give a combinatorial algorithm for generalized network flow problems under separable convex cost functions. It is easy to see that **ADNB** can be cast in their framework; however, their algorithm does not run in polynomial time and moreover it assumes a feasible solution to start off. On the other hand, one of the main points of our algorithm is testing feasibility.

2 Rational Convex Programs

A nonlinear convex program is said to be *rational* if, for any setting of its parameters to rational numbers such that it has a finite optimal solution, it admits an optimal solution that is rational and can be written using polynomially many bits in the number of bits needed to write all the parameters[4]. We will abbreviate the name to RCP.

How is it that despite its nonlinearity, an RCP always has a rational optimal solution? The answer lies in studying KKT conditions, which characterize optimality for a large class of convex programs. For this purpose, consider the following general form of a convex program, where $f_0, f_1, \ldots, f_m$ are convex functions and the second set of constraints are affine.

$$\text{minimize} \quad f_0(x) \tag{1}$$
$$\text{subject to} \quad \forall i, \ 1 \le i \le m: \ f_i(x) \le 0$$

---

[3]Stage I can also be accomplished quite easily via the use of an LP-solver. Feasibility can be tested by solving the LP (5), and one of the referees of this paper has given a clever LP that finds feasible prices as well, in case the instance is feasile. However, our emphasis is not on a practical solution with commercial solvers but rather on obtaining a fully combinatorial algorithm and studying the combinatorial aspects of the problem in depth.

[4]Strictly speaking, a more general definition is called for which considers a countably infinite set of "well-formed" convex programs, and this definition can be made precise by assuming a polynomial time procedure that determines whether a given program is "well-formed". We have avoided this level of generality for the sake of simplicity; however, the reader should be able to easily reconstruct it.

$$\forall j, \ 1 \le j \le p: \quad {a_i}^T x = b_i$$

Clearly, the optimal solution must satisfy all the constraints. In addition, the KKT conditions involve a equations containing partial derivatives of the objective. Hence, one way of obtaining an RCP is to ensure that all constraints are linear and the derivative of the objective can be written as a linear function. This gives two classes of RCPs: quadratic and logarithmic. (Intuitively, the latter case works because the derivative of $log x$ is $1/x$, and sometimes, we can replace $1/x$ by a new variable, say y.) Are there more classes of RCPs? We leave this as an important open question.

Assume that in program (1), the functions $f_1, \ldots f_m$ are all linear and that f_0 is the quadratic function $x^T P x + q^T x$. Now, f_0 will be convex if and only if its Hessian, $\nabla^2 f_0 \succeq 0$, i.e., if and only if P is a positive semidefinite matrix. If so, program (1) will be an RCP. This gives us the class of *quadratic RCPs*. Such programs have numerous important applications, such as constrained regression or constrained least-squares (see Chapter 4 of [BV04]) and there have been attempts at designing efficient algorithms, e.g., see [LFB06] for special algorithms for portfolio optimization.

Next, assume that in program (1), the functions $f_1, \ldots f_m$ are all linear and that f_0 is the logarithmic function

$$f_0(x) = -\sum_{i=1}^{n} c_i \log g_i(x),$$

where $g_1, \ldots, g_n$ are linear functions and c_i's are constants. Clearly, this is a convex program; however, it is not always rational. For specific choices of the functions $f_1, \ldots f_m$ and $g_1, \ldots, g_n$, it turns out to be an RCP, and this needs to be established piecemeal. One such proof of rationality is provided in Theorem 2. We will call this class *logarithmic RCPs*.

The following strongly polynomial algorithms, under Program A, have been given for solving specific quadratic RCPs. To the best of our knowledge, the first such result was due to Helgason, Kennington, Lall [HKL80], who gave an algorithm for the very special case of a single equality constraint $\sum_j x_j = c$, together with non-negativity and upper bounds on the x_j's. Minoux [Min84] extended to minimum quadratic cost flow problems. Frank and Karzanov [FK92] solved the problem of finding the closest point, from the origin, to the perfect matching polytope of a given bipartite graph. Hochbaum and Shantikumar [HS90] and Karzanov and McCormick [KM97] generalized all the previous problems by solving the problem of minimizing an arbitrary quadratic function subject to the system $Ax = 0$, where A is an arbitrary totally unimodular matrix. An arbitrary quadratic RCP can be solved in polynomial time by using an LP-oracle, i.e., Program B, using an idea of Ben-Tal and Nemirovski [BTN01], given in the context of polyhedral approximations of the second order cone. This idea shows how to write a (weakly) polynomially long LP whose optimal solution gives an answer to the given instance. Obtaining a strongly polynomial algorithm, which uses an LP oracle, remains an important open question.

For solving specific logarithmic RCPs, the following polynomial algorithms, under Program A, have been given. The Eisenberg-Gale program, which captures equilibrium for linear Fisher markets is solved in [DPSV08] and a strongly polynomial algorithm was given by Orlin

[Orl10]. Other Fisher markets include utility functions defined via combinatorial problems [JV08], including some in Kelly's [Kel97] resource allocation model, spending constraint utilities [Vaz10, BDX10], and piecewise-linear concave utilities in a market model that allows for perfect price discrimination [GV11]. The following papers give algorithms using an LP-oracle, i.e., Program B: Eisenberg-Gale markets with 2 buyers [CDV10] and Nash bargaining games with 2 agents [Vaz].

As in the case of ILPs, insights gained from combinatorial algorithms for RCPs have also led to major progress – this includes definitions of several of the models and RCPs described in the previous paragraph. As another instance, the recent proof of membership in PPAD of markets under piecewise-linear concave utilities [VY11] followed from a new, combinatorial way of characterizing equilibria [DPSV08] and helped settle, together with [CDDT09, CT09], the long-standing open problem of determining the exact complexity of this key market model.

Combinatorial algorithms also have several advantages over continuous algorithms in applications. For instance, recently Nisan et. al., faced with the problem of designing an auction system for Google for TV ads, converged to a market equilibrium based method, after exploring several different options [NBC+09]. As stated by Nisan [Nis09], the actual implementation of this algorithm was inspired by combinatorial market equilibrium algorithms, which in turn solve convex programs combinatorially. Indeed, the easy adaptability of combinatorial algorithms to the special idiosyncrasies of an application often makes them the preferred method. In Section 5 we present an application of our combinatorial algorithm for **ADNB** to a throughput allocation problem on a wireless channel.

3   Nash Bargaining Games

An *n-person Nash bargaining game* consists of a pair $(\mathcal{N}, \boldsymbol{c})$, where $\mathcal{N} \subseteq \mathbf{R}_+^n$ is a compact, convex set and $\boldsymbol{c} \in \mathcal{N}$. Set $\mathcal{N}$ is the *feasible set* and its elements give utilities that the n players can simultaneously accrue. Point $\boldsymbol{c}$ is the *disagreement point* – it gives the utilities that the n players obtain if they decide not to cooperate. The set of n agents will be denoted by B and the agents will be numbered $1, 2, \ldots n$. Game $(\mathcal{N}, \boldsymbol{c})$ is said to be *feasible* if there is a point $\boldsymbol{v} \in \mathcal{N}$ such that $\forall i \in B$, $v_i > c_i$, and *infeasible* otherwise.

The solution to a feasible game is the point $\boldsymbol{v} \in \mathcal{N}$ that satisfies the following four axioms:

1. **Pareto optimality:** No point in $\mathcal{N}$ can weakly dominate $\boldsymbol{v}$.

2. **Invariance under affine transformations of utilities:** If the utilities of any player are redefined by multiplying by a scalar and adding a constant, then the solution to the transformed game is obtained by applying these operations to the particular coordinate of $\boldsymbol{v}$.

3. **Symmetry:** If the players are renumbered, then it suffices to renumber the coordinates of $\boldsymbol{v}$ accordingly.

4. **Independence of irrelevant alternatives:** If $\boldsymbol{v}$ is the solution for $(\mathcal{N}, \boldsymbol{c})$, and $\mathcal{S} \subseteq \mathbf{R}_+^n$ is a compact, convex set satisfying $\boldsymbol{c} \in \mathcal{S}$ and $\boldsymbol{v} \in \mathcal{S} \subseteq \mathcal{N}$, then $\boldsymbol{v}$ is also the solution for $(\mathcal{S}, \boldsymbol{c})$.

Via an elegant proof, Nash proved:

Theorem 1 Nash *[Nas50] If game $(\mathcal{N}, \boldsymbol{c})$ is feasible then there is a unique point in $\mathcal{N}$ satisfying the axioms stated above. This is also the unique point that maximizes $\Pi_{i \in B}(v_i - c_i)$, over all $\boldsymbol{v} \in \mathcal{N}$.*

Most papers in game theory assume that the given Nash bargaining game $(\mathcal{N}, \boldsymbol{c})$ is feasible. However, in this paper, it will be more natural to not make this assumption and to determine this fact algorithmically. Henceforth, we will drop the assumption that the given game is feasible.

Thus Nash's solution to his bargaining game involves maximizing a concave function over a convex domain, and is therefore the optimal solution to the following convex program.

$$\text{maximize} \quad \sum_{i \in B} \log(v_i - c_i) \tag{2}$$
$$\text{subject to} \quad \boldsymbol{v} \in \mathcal{N}.$$

As a consequence, if for a specific game, a separation oracle can be implemented in polynomial time, then using the ellipsoid algorithm one can get as good an approximation to the solution of this convex program as desired in time polynomial in the number of bits of accuracy needed [GLS88].

4 The Game ADNB

The game **ADNB**, short for *Arrow-Debreu Nash Bargaining game*, is derived from the linear case of the Arrow-Debreu model.

We first state formally the linear case of the Arrow-Debreu model. Let $B = \{1, 2, \ldots, n\}$ be a set of agents and $G = \{1, 2, \ldots, g\}$ be a set of divisible goods. We will assume w.l.o.g. that there is a unit amount of each good. Let u_{ij} be the utility derived by agent i on receiving one unit of good j. We will assume that u_{ij} is integral; this is w.l.o.g. for proving existence of a polynomial time algorithm since multiplying all utilities by the least common multiple of the denominators leaves the problem unchanged and preserves polynomial running time. If x_{ij} is the amount of good j that agent i gets, for $1 \leq j \leq g$, then she derives total utility

$$v_i(x) = \sum_{j \in G} u_{ij} x_{ij}.$$

Finally, we assume that each agent has an initial endowment of these goods; for each good, the total amount possessed by the agents is 1 unit.

W.l.o.g. we may assume that each good is desired by at least one agent and each agent desires at least one good, i.e.,

$$\forall j \in G, \ \exists i \in B : \ u_{ij} > 0 \ \text{ and } \ \forall i \in B, \ \exists j \in G : \ u_{ij} > 0.$$

If not, we can remove the good or the agent from consideration.

The question is to find prices for these goods so that if each agent sells her entire initial endowment at these prices and uses the money to buy an optimal bundle of goods, the market clears exactly, i.e., there is no deficiency or surplus of any good. Such prices are called *equilibrium prices*.

The Arrow-Debreu market model gives one mechanism by which the agents can redistribute goods to achieve higher utilities. Another mechanism is to view this setup as a Nash bargaining game as follows. For each $i \in B$, let c_i denote the utility derived by agent i from her initial endowment; again, w.l.o.g. we will assume that c_i is integral. Regard this as agent i's disagreement utility and redistribute the goods in accordance with the Nash bargaining solution.

We next define the *Arrow-Debreu Nash Bargaining game*. The setup is different from above only in that in the given instance, instead of initial endowments of agents, we are specified disagreement utilities, c_is, which are arbitrary non-negative numbers. The Nash bargaining solution to this instance is the optimal solution to the following convex program:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i \in B} \log(v_i - c_i) && (3) \\
\text{subject to} \quad & \forall i \in B: \ v_i = \sum_{j \in G} u_{ij} x_{ij} \\
& \forall j \in G: \ \sum_{i \in B} x_{ij} \leq 1 \\
& \forall i \in B, \ \forall j \in G: \ x_{ij} \geq 0
\end{aligned}
$$

Let p_j be the Lagrange variable corresponding to the inequality constraint in this program. Note that since the objective is strictly concave, the dual $\boldsymbol{p}$ will be unique. The KKT conditions for this program are:

(1) $\forall j \in G: \ p_j \geq 0$.

(2) $\forall j \in G: \ p_j > 0 \Rightarrow \sum_{i \in B} x_{ij} = 1$.

(3) $\forall i \in B, \ \forall j \in G: p_j \geq \frac{u_{ij}}{v_i - c_i}$.

(4) $\forall i \in B, \ \forall j \in G: x_{ij} > 0 \Rightarrow p_j = \frac{u_{ij}}{v_i - c_i}$.

Theorem 2 *Program (3) is a rational convex program. Moreover, if it is feasible, then the dual solution is unique.*

Proof : We will show that there is a exponential family of LPs, each with rational parameters, one of which provide the solution to program (3); hence the solution is rational and can be written using polynomially many bits. Observe that this argument will also show that the feasibility problem lies in the class NP.

Guess the x_{ij}'s that are non-zero in the optimal solution to program (3). By the assumption made on the instance, each p_j will be positive. Each guess will give one LP: The variables of the LP will be the non-zero x_{ij}'s and for each good j, a new variable q_j, which is supposed to represent $1/p_j$. The LP will have the following constraints: for each q_j, there is one equation corresponding to the KKT condition (2), and for each nonzero x_{ij} there is one equation corresponding to the KKT condition (4). In addition, the LP has inequality constraints corresponding to KKT condition (3), for each $i \in B$ and $j \in G$. In all these constraints, v_i is replaced by $\sum_{j \in G} u_{ij} x_{ij}$. Finally, it has non-negativity constraints for all x_{ij}'s and q_j's. It is easy to check that all constraints are linear.

Since program (3) is feasible, so is the LP corresponding to the correct guess and yield an optimal solution to program (3). Strict concavity of the objective function of program (3) implies that the optimal values of v_i's is unique. Now, using the KKT condition (4), we get the uniqueness of p_j's as well. □

Theorem 2 establishes "granularity" in **ADNB** which is crucial for establishing an upper bound on the running time of our algorithm. For this purpose, we give the following crucial definition.

Definition : Let us define μ to be an upper bound on the largest denominator that can arise in the rational solution to program (3). Define $U = \max_{i \in B, j \in G} \{u_{ij}\}$. By the proof of Theorem 2, $\mu \leq U^{g(n+1)}$. □

5 An Application to Throughput Allocation on a Wireless Channel

A central throughput allocation problem arising in the context of a wireless channel, such as in 3G technologies, is the following. There are n users $1, 2, \ldots n$, and the wireless router can be in any of m different states $1, 2, \ldots m$ whose probabilities, $\pi(j)$, can be estimated by sampling. Each user i derives utility at rate u_{ij} if it is connected to the router while the router is in state j; the u_{ij}'s are known. No matter what state the router is in, only one user can be connected to it. If user i is given connection for $x_{ij} \leq \pi(j)$ of the time the router is in state j, for $1 \leq j \leq m$, then the total utility derived by i is $v_i = \sum_{j=1}^{m} u_{ij} x_{ij}$. Clearly, we must ensure the constraint $\sum_{i=1}^{n} x_{ij} \leq \pi(j)$, for each j. The question is to find a "fair" way of dividing the $\pi(j)$'s among the users.

The method of choice in the networking community is to use Kelly's proportional fair scheme [Kel97], which entails maximizing $\sum_i \log v_i$ subject to the constraints given above, i.e., solving the Eisenberg-Gale convex program [EG59]. Observe that the above setting can be viewed as a linear Fisher market with n users and m divisible goods. An elegant gradient descent algorithm for solving this convex program, given by David Tse [Tse] (see also [JPP00]), was implemented by Qualcomm in their chip sets and is used by numerous 3G wireless base stations [And09]. However, this solution may at times allocate unacceptably low utility to certain users. This was countered by giving users the ability to put a lower bound on channel rates, say c_i for user i. This enhanced problem was solved by changing the objective function of the convex

program to maximizing $\sum_i \log(v_i - c_i)$; observe that this is precisely an instance of **ADNB**! However, now the gradient descent implementation ran into problems of instability, since it involved computing $u_{ij}/(v_i' - c_i)$, where v_i' is the current estimate of v_i; at intermediate points, the denominator may be too small or even negative.

A different solution, proposed and implemented by researchers at Lucent [AQS05], was to introduce the constraints $v_i > c_i$ in the Eisenberg-Gale program itself. The fairness guarantee achieved by this solution is unclear. Additionally, determining feasibility of the convex program now became a major issue [And09]. Instead, we have proposed experimenting with a heuristic adaptation of our combinatorial algorithm for **ADNB**, which will not have stability issues. As reported in the FOCS 2002 version of [DPSV08], an analogous heuristic adaptation of the DPSV algorithm was found to perform well on fairly large sized linear Fisher instances.

6 Fisher's Model and its Extension via Flexible Budgets

First we specify Fisher's market model for the case of linear utilities [BS00]. Consider a market consisting of a set of n buyers $B = \{1, 2, \ldots, n\}$, and a set of g divisible goods, $G = \{1, 2, \ldots, g\}$; we may assume w.l.o.g. that there is a unit amount of each good. Let m_i be the money possessed by buyer i, $i \in B$. Let u_{ij} be the utility derived by buyer i on receiving one unit of good j. Thus, if x_{ij} is the amount of good j that buyer i gets, for $1 \le j \le g$, then the total utility derived by i is

$$v_i(x) = \sum_{j=1}^{g} u_{ij} x_{ij}.$$

The problem is to find prices $\boldsymbol{p} = \{p_1, p_2, \ldots, p_g\}$ for the goods so that when each buyer is given her utility maximizing bundle of goods, the market clears, i.e., each good having a positive price is exactly sold, without there being any deficiency or surplus. Such prices are called *market clearing prices* or *equilibrium prices*.

The following is the Eisenberg-Gale convex program. Using the KKT conditions, one can show that its optimal solution is an equilibrium allocation for Fisher's linear market and the Lagrange variables corresponding to the inequalities give equilibrium prices for the goods (e.g., see Theorem 5.1 in [Vaz07]).

$$\text{maximize} \quad \sum_{i \in B} m_i \log v_i \tag{4}$$
$$\text{subject to} \quad \forall i \in B: \ v_i = \sum_{j \in G} u_{ij} x_{ij}$$
$$\forall j \in G: \ \sum_{i \in B} x_{ij} \le 1$$
$$\forall i \in B, \ \forall j \in G: \ x_{ij} \ge 0$$

Next, we introduce a flexible budget market as a modification of Fisher's linear case; this market will be used for solving **ADNB**. The utility functions of buyers are as before. The two

main differences are that each buyer i now has a parameter c_i giving a strict lower bound on the amount of utility she wants to derive, and buyers do not come to the market with a fixed amount of money, but instead the money they spend is a function of prices of goods in the following manner.

Let us convince the reader that the notion of a flexible budget market is a natural one. Suppose the goods are different foods u_{ij} represents the number of calories agent i derives from 1 unit of good j. Assume that c_i is a strict lower bound on the total number of calories agent i wants from her bundle. To achieve this, she is willing to spend 1 dollar more than the minimum money needed to get c_i calories. The problem is to find prices such that the market clears. Clearly, this is a flexible budget market.

Definition :   Given prices $\boldsymbol{p}$ for the goods, define the *maximum bang-per-buck* of buyer i to be

$$\gamma_i = max_j \left\{ \frac{u_{ij}}{p_j} \right\}.$$

We will say that set $S_i = \operatorname{argmax}_j \left\{ \frac{u_{ij}}{p_j} \right\}$ constitutes i's *maximum bang-per-buck goods*. Now, *buyer i's money* is defined to be $m_i = 1 + \frac{c_i}{\gamma_i}$. We will denote $\frac{c_i}{\gamma_i}$ by α_i.   □

Clearly, at prices $\boldsymbol{p}$, any utility maximizing bundle of goods for i will consist of goods from S_i costing m_i money. Again the problem is to find market clearing or equilibrium prices. Observe that in an equilibrium, if it exists, each buyer i will derive utility exceeding c_i.

6.1   The Reduction

An instance I of **ADNB** is transformed to a flexible budget market $\mathcal{M}$ as stated above.

Theorem 3 *Instance I of **ADNB** is feasible if and only if there is a feasible solution for the corresponding flexible budget market. Moreover, if I and $\mathcal{M}$ are both feasible, then allocations $\boldsymbol{x}$ and dual $\boldsymbol{p}$ are optimal for I if and only if they are equilibrium allocations and prices for the flexible budget market $\mathcal{M}$.*

Proof :   We will use the fact that since convex program (3) has a concave objective and linear constraints, the KKT conditions are both necessary and sufficient for optimality.

($\Rightarrow$) First assume that I is feasible and that allocations $\boldsymbol{x}$ and dual $\boldsymbol{p}$ are optimal for **ADNB** game I. Then I must satisfy the KKT conditions for convex program (3).

By the second KKT condition, each good having a positive price is fully sold. Assume that $x_{ij} > 0$. Then, by the definition of γ_i and the fourth KKT condition,

$$\gamma_i = \frac{u_{ij}}{p_j} = v_i - c_i.$$

The money of buyer i at prices $\boldsymbol{p}$ in market $\mathcal{M}$ is defined to be $m_i = 1 + c_i/\gamma_i$. The money spent by i in market $\mathcal{M}$ is:

$$\sum_{j \in G} x_{ij} p_j = \sum_{j \in G} \frac{x_{ij} u_{ij}}{\gamma_i}$$

$$= \frac{1}{v_i - c_i} \sum_{j \in G} x_{ij} u_{ij} = \frac{v_i}{v_i - c_i} = 1 + \frac{c_i}{v_i - c_i} = 1 + \frac{c_i}{\gamma_i} = m_i.$$

Furthermore, by the third and fourth KKT conditions, i buys only her maximum bang-per-buck objects, thereby getting an optimal bundle. This proves that $\boldsymbol{x}$ and $\boldsymbol{p}$ constitute equilibrium allocations and prices for market $\mathcal{M}$.

($\Leftarrow$) Next, assume that $\mathcal{M}$ is feasible and that $\boldsymbol{x}$ and $\boldsymbol{p}$ are equilibrium allocations and prices for market $\mathcal{M}$. Now, $\boldsymbol{x}$ is clearly feasible for program (3); we will show that $\boldsymbol{x}$ and $\boldsymbol{p}$ satisfy all the KKT conditions for this program. The first two conditions are obvious.

Since i gets an optimal bundle of objects at prices $\boldsymbol{p}$,

$$x_{ij} > 0 \;\Rightarrow\; \frac{u_{ij}}{p_j} = \gamma_i.$$

Since i spends all her money,

$$m_i = 1 + \frac{c_i}{\gamma_i} = \sum_{j \in G} x_{ij} p_j = \sum_{k \in T_i} x_{ik} \frac{u_{ik}}{\gamma_i} = \frac{v_i}{\gamma_i}.$$

Therefore, $\gamma_i = v_i - c_i$. This gives the last two conditions as well. $\qquad\square$

By the KKT conditions, if convex program (3) is feasible, its dual is unique. Hence, in this case, market $\mathcal{M}$ will have unique equilibrium prices.

7   Some Properties of Equilibrium Prices

Our goal is to present an efficient algorithm for solving an instance I of the game **ADNB** by first reducing it to a flexible budget market $\mathcal{M}$, say. First we give an efficient algorithm for the following simpler question: Given prices $\boldsymbol{p} = \{p_1, \ldots p_g\}$ for the goods in $\mathcal{M}$, determine if these are equilibrium prices, and if so, find an equilibrium allocation.

The algorithm begins by constructing a directed network $N(\boldsymbol{p})$ as follows. $N(\boldsymbol{p})$ has a source s, a sink t, and vertex subsets B and G corresponding to the buyers and goods, respectively. For each good $j \in G$, there is an edge (s, j) of capacity p_j, and for each buyer $i \in B$, there is an edge (i, t) of capacity m_i, where $m_i = 1 + c_i/\gamma_i$ is i's money in $\mathcal{M}$. Recall that S_i contains i's maximum bang-per-buck goods. The edges between G and B are precisely the maximum bang-per-buck edges, i.e., those (j, i) such that $j \in S_i$. Each of these edges has infinite capacity.

Lemma 4 *Prices $\boldsymbol{p}$ are equilibrium prices for $\mathcal{M}$ if and only if the two cuts $(s, B \cup G \cup t)$ and $(s \cup B \cup G, t)$ are min-cuts in network $N(\boldsymbol{p})$. Moreover, if $\boldsymbol{p}$ are equilibrium prices, then the set of equilibrium allocations corresponds exactly to max-flows in $N(\boldsymbol{p})$.*

The proof of this lemma is straightforward using the transformation between a max-flow f in $N(\boldsymbol{p})$ and an allocation x in $\mathcal{M}$ given by $x_{ij} = f(j,i)/p_j$. The condition that $(s, B \cup G \cup t)$ and $(s \cup B \cup G, t)$ are min-cuts in network $N(\boldsymbol{p})$, and hence saturated by f, corresponds to all goods being sold and all buyers' money being spent. The fact that (j,i) is an edge in $N(\boldsymbol{p})$ if and only if $j \in S_i$ ensures that buyers get only their maximum bang-per-buck goods. Clearly, one max-flow computation suffices to determine if prices $\boldsymbol{p}$ are equilibrium prices for $\mathcal{M}$.

Above, we have shown how to derive H and equilibrium allocations from prices $\boldsymbol{p}^*$. Next, we show how to determine $\boldsymbol{p}^*$ from H efficiently, thereby showing that H is the combinatorial object that yields equilibrium prices. Assume that $\boldsymbol{p}^*$ are equilibrium prices, i.e., $N(\boldsymbol{p}^*)$ satisfies the condition in Lemma 4. Let H be the uncapacitated directed subgraph of $N(\boldsymbol{p}^*)$ induced on $B \cup G$.

Lemma 5 *Given H, $\boldsymbol{p}^*$ can be computed in strongly polynomial time.*

Proof : Consider the connected components of H after ignoring directions on its edges. In each component, pick a good and assign it price p, say. The prices of the rest of the goods in this component can be obtained in terms of p. The bang-per-buck, and hence the money, of each buyer in this component can also be obtained in terms of p. Finally, by equating the money of all buyers in this component with the total value of all goods in this component, we can compute p. □

Given two d-dimensional vectors with non-negative coordinates, $\boldsymbol{p}$ and $\boldsymbol{q}$, we will say that $\boldsymbol{p}$ *weakly dominates* $\boldsymbol{q}$ if for each coordinate j, $q_j \leq p_j$. The following characterization will be useful in Algorithm 3.

Lemma 6 *Assume that market $\mathcal{M}$ is feasible and $\boldsymbol{p}$ is its unique equilibrium price vector. Let $\boldsymbol{q}$ be a vector of positive prices such that $(s, B \cup G \cup t)$ is a min-cut in network $N(\boldsymbol{q})$. Then $\boldsymbol{p}$ weakly dominates $\boldsymbol{q}$.*

Proof : Let us assume, for establishing a contradiction, that there are goods j such that $q_j > p_j$ and yet $(s, B \cup G \cup t)$ is a min-cut in $N(\boldsymbol{q})$. Let

$$\theta = \max_{j \in G}\left\{ \frac{q_j}{p_j} \right\} \quad \text{and} \quad S = \{j \in G \mid q_j = \theta p_j\}.$$

Clearly, $\theta > 1$.

Let T_p and T_q be the set of buyers who are interested in goods in S at prices p and q, respectively. Since S represents the set of goods whose prices increase by the largest factor in going from prices $\boldsymbol{p}$ to $\boldsymbol{q}$, $T_q \subseteq T_p$. We claim that a buyer $i \in T_q$ is not interested in any goods in $G - S$ at prices $\boldsymbol{p}$ (because otherwise at prices $\boldsymbol{q}$, i will not be interested in any goods in S, since their prices increased the most). Therefore, in any max-flow in $N(\boldsymbol{p})$, all flow going through nodes in T_q must also have used nodes in S. Therefore,

$$\sum_{j \in S} p_j \geq \sum_{i \in T_q} \left(1 + \frac{c_i}{\gamma_i} \right),$$

14

where γ_i is the maximum bang-per-buck of buyer i w.r.t. prices $\boldsymbol{p}$. Multiplying this inequality by θ and using the fact that $\theta > 1$ we get,

$$\theta \sum_{j \in S} p_j \geq \sum_{i \in T_q} \left(\theta + \frac{c_i \theta}{\gamma_i} \right) > \sum_{i \in T_q} \left(1 + \frac{c_i \theta}{\gamma_i} \right),$$

Observe that the maximum bang-per-buck of buyer $i \in T_q$ w.r.t. prices $\boldsymbol{q}$ is γ_i / θ. Therefore, the last inequality implies that w.r.t. prices $\boldsymbol{q}$, the total value of goods in S is strictly more than the total value of money possessed by buyers in T_q. On the other hand, since in $N(\boldsymbol{q})$ all flow using nodes of T_q goes through S, we get that $(s, B \cup G \cup t)$ is not a min-cut in network $N(\boldsymbol{q})$, leading to a contradiction. □

Next assume that $\mathcal{M}$ is an arbitrary flexible budget market, not necessarily feasible. We will say that prices $\boldsymbol{q}$ are *small* if $\boldsymbol{q}$ is a positive vector and $(s, B \cup G \cup t)$ is a min-cut in network $N(\boldsymbol{q})$. Observe that in this case, each good j must have an edge (j, i), for some buyer i, incident at it. By Lemma 6, if $\mathcal{M}$ is feasible and prices $\boldsymbol{q}$ are small, then they are weakly dominated by the equilibrium prices, $\boldsymbol{p}$. Observe however that the contrapositive of Lemma 6 does not hold, i.e., $\boldsymbol{p}$ may dominate positive prices $\boldsymbol{q}$, yet $(s, B \cup G \cup t)$ may not be a min-cut in network $N(\boldsymbol{q})$.

The background given so far will suffice to read Section A, which gives an algorithm that converges to the solution of a given feasible instance of **ADNB** in the limit. This section may also be viewed as a warm-up for the polynomial time algorithm, which is quite involved.

8   Characterizing Feasibility and Infeasibility

In this section, we will address the question of determining whether the given flexible budget market, $\mathcal{M}$, is feasible. We will give a characterization of feasible markets and we will derive conditions that yield a proof of infeasibility.

Clearly, if we can find small prices $\boldsymbol{p}$ and a max-flow f in network $N(\boldsymbol{p})$ such that the flow gives each buyer i strictly more than c_i utility, then $\mathcal{M}$ is feasible. We first show, using the notion of balanced flows, that this test of feasibility is in fact a property of prices $\boldsymbol{p}$ only.

8.1   Balanced flows

We will follow the exposition in [Vaz07] and refer the reader to this chapter for all facts stated below without proof. For simplicity, let N denote the current network, $N(\boldsymbol{p})$. Given a feasible flow f in N, let $R(f)$ denote the residual graph w.r.t. f. Define the *surplus* of buyer i w.r.t. flow f in network N, $\theta_i(N, f)$, to be the residual capacity of the edge (i, t) w.r.t. flow f in network N, i.e., m_i minus the flow sent through the edge (i, t). The *surplus vector w.r.t. flow* f is defined to be $\theta(N, f) := (\theta_1(N, f), \theta_2(N, f), \ldots, \theta_n(N, f))$. Let $\|v\|$ denote the l_2 norm of vector v. A *balanced flow* in network N is a flow that minimizes $\|\theta(N, f)\|$. A balanced flow must be a max-flow in N because augmenting a given flow can only lead to a decrease in the l_2 norm of the surplus vector.

A balanced flow in N can be computed using at most n max-flow computations. It is easy to see that all balanced flows in N have the same surplus vector. Hence, for each buyer i, we can define $\theta_i(N)$ to be the surplus of i w.r.t. any balanced flow in N; we will shorten this to θ_i when the network is understood. The key property of a balanced flow that our algorithm will rely on is that a maximum flow f in N is balanced if and only if it satisfies Property 1:

Property 1: For any two buyers i and j, if $\theta_i(N, f) < \theta_j(N, f)$ then there is no path from node i to node j in $R(f) - \{s, t\}$.

Balanced flows play a crucial role in both stages of our algorithm; moreover, they have multiple uses. In Section 10.4, after stating the full algorithm, we state the various uses of this notion.

8.2 A characterization of feasibility

Let $\boldsymbol{p}$ be small prices and let $(\theta_1, \ldots, \theta_n)$ be the surplus vector of a balanced flow in $N(\boldsymbol{p})$. We will say that $\boldsymbol{p}$ are *feasible prices* if for each buyer i, $\theta_i < 1$.

Lemma 7 *Market $\mathcal{M}$ is feasible if and only if it admits feasible prices.*

Proof :   If $\mathcal{M}$ is feasible, its equilibrium prices are feasible, since for each buyer i, $\theta_i = 0$. Next, assume that $\boldsymbol{p}$ are feasible prices for $\mathcal{M}$. By definition, the flow sent on edge (i, t) in a balanced flow in $N(\boldsymbol{p})$ is

$$m_i - \theta_i > m_i - 1 = \left(1 + \frac{c_i}{\gamma_i}\right) - 1 = \frac{c_i}{\gamma_i}.$$

The utility accrued by i from this allocation is $\gamma_i(m_i - \theta_i) > c_i$. Hence $\mathcal{M}$ is feasible.   □

Observe that if a max-flow f in network $N(\boldsymbol{p})$ gives each buyer i strictly more than c_i utility, then so will a balanced flow in $N(\boldsymbol{p})$. Hence, feasibility is a property of prices $\boldsymbol{p}$ only.

Rather than working with θ_i, it will sometimes be more convenient to work with $\theta_i - 1$.

Definition :   Let $\boldsymbol{p}$ be small prices. W.r.t. these prices, define the *1-surplus* of buyer i to be $\beta_i = \theta_i - 1$.   □

Now, another definition of feasible prices is that they be small and for each buyer i, $\beta_i < 0$.

8.3 A characterization of infeasibility

We will establish infeasibility of the given market $\mathcal{N}$ by using the dual of an LP that tests for feasibility of $\mathcal{M}$. As defined in Section 3, the given game is feasible if and only if there is a point $v \in \mathcal{N}$ such that for each agent $i \in B$, $v_i > c_i$. In order to capture feasibility via a linear program, let us restate as follows: an instance of **ADNB** is feasible if and only if

$$\max_{v \in \mathcal{N}} \ \min_{i \in B} \{v_i - c_i\} > 0.$$

By Theorem 2, an instance of **ADNB** is feasible if and only if

$$\max_{v \in \mathcal{N}} \min_{i \in B} \{v_i - c_i\} \geq \frac{1}{\mu}.$$

Observe that the expression on the left hand side is the optimal objective function value of LP (5). Clearly, this LP is maximizing t; however, in order to obtain a convenient dual, we will write it as minimizing $-t$:

$$
\begin{aligned}
\text{minimize} \quad & -t && (5) \\
\text{subject to} \quad & \forall i \in B: \ \sum_{j \in G} u_{ij} x_{ij} \geq c_i + t \\
& \forall j \in G: \ -\sum_{i \in B} x_{ij} \geq -1 \\
& \forall i \in B, \ \forall j \in G: \ x_{ij} \geq 0
\end{aligned}
$$

Let y_i's and z_j's be the dual variables corresponding to the first and second set of inequalities, respectively. The dual program is:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i \in I} c_i y_i - \sum_{j \in G} z_j && (6) \\
\text{subject to} \quad & \forall i \in B, \ \forall j \in G: \ u_{ij} y_i - z_j \leq 0 \\
& \sum_{i \in B} y_i = 1 \\
& \forall i \in B: \ y_i \geq 0 \\
& \forall j \in G: \ z_j \geq 0
\end{aligned}
$$

In Lemma 8, we will establish a useful fact. For agent i, denote c_i/γ_i by α_i. Hence, $m_i = 1 + \alpha_i$.

Lemma 8
$$\sum_{i \in B} \alpha_i - \sum_{j \in G} p_j = \sum_{i \in B} \beta_i.$$

Proof : The total surplus of all buyers is

$$\sum_{i \in B} \theta_i = \sum_{i \in B} m_i - \sum_{j \in G} p_j = \sum_{i \in B} (1 + \alpha_i) - \sum_{j \in G} p_j = n + \sum_{i \in B} \alpha_i - \sum_{j \in G} p_j.$$

On the other hand,

$$\sum_{i \in B} \theta_i = \sum_{i \in B} (1 + \beta_i) = n + \sum_{i \in B} \beta_i.$$

Equating the two we get the lemma. □

17

Lemma 9 (Main Lemma) *If there exist prices $\boldsymbol{p}$ s.t.*

$$n \geq \sum_{j \in G} p_j > 0 \quad and \quad \sum_{i \in B} \beta_i \geq -\frac{n^2}{\mu},$$

then market $\mathcal{M}$ is infeasible.

Proof : We will consider 2 cases, by splitting the range $[-n^2/\mu, \infty)$ into the 2 ranges $[0, \infty)$ and $[-n^2/\mu, 0)$; the former will be called Case 1 and the latter Case 2. For the first case will use LP (5) and for the second case we will modify this LP by adding a non-negativity condition on t.

For each buyer i, let γ_i denote the maximum bang-per-buck of buyer i w.r.t. prices $\boldsymbol{p}$. By definition of maximum bang-per-buck,

$$\forall i \in B, \ \forall j \in G : \quad \gamma_i \geq \frac{u_{ij}}{p_j}.$$

Case 1: Let $\nu = \sum_{i \in B} 1/\gamma_i$. Since $\sum_{j \in G} p_j > 0$, $\nu > 0$. Consider prices $\boldsymbol{q}$, where for each $j \in G$, $q_j = p_j/\nu$. Since all the prices have been scaled by the same factor, the network remains unchanged. Clearly, the maximum bang-per-buck of buyer i w.r.t. $\boldsymbol{q}$ is $\gamma_i' = \nu \gamma_i$ and

$$\forall i \in B, \ \forall j \in G : \quad \gamma_i' \geq \frac{u_{ij}}{q_j}.$$

Let $y_i = 1/\gamma_i'$, for $i \in B$, and $z_j = q_j$, for $j \in G$. We will show that (y, z) is a feasible solution for the dual LP (6). The first set of inequalities is established by noting that

$$\forall i \in B, \ \forall j \in G : \quad \gamma_i' \geq \frac{u_{ij}}{q_j} \quad \text{hence} \quad u_{ij} y_i \leq z_j.$$

Next we show that the equality constraint holds:

$$\sum_{i \in B} y_i = \sum_{i \in B} \frac{1}{\gamma_i'} = \left(\frac{1}{\nu}\right) \cdot \sum_{i \in B} \frac{1}{\gamma_i} = 1.$$

Let $\alpha_i' = c_i/\gamma_i'$ and let β_i' be the 1-surplus of buyer i w.r.t. prices $\boldsymbol{q}$. The objective function value of the dual solution (y, z) is

$$\sum_{i \in B} c_i y_i - \sum_{j \in G} z_j = \sum_{i \in B} \frac{c_i}{\gamma_i'} - \sum_{j \in G} z_j = \sum_{i \in B} \alpha_i' - \sum_{j \in G} q_j = \sum_{i \in B} \beta_i' = \left(\frac{1}{\nu}\right) \cdot \sum_{i \in B} \beta_i \geq 0.$$

We have used Lemma 8 in the third equality. Therefore, at optimality, $-t \geq \sum_{i \in B} c_i y_i - \sum_{j \in G} z_j \geq 0$, i.e., $t < 0$, hence establishing infeasibility of the game.

Case 2: For the second range, it suffices to show a feasible solution of value at least $-1/\mu$ to the dual of the following LP:

$$\text{minimize} \quad -t \tag{7}$$

$$\text{subject to} \quad \forall i \in B: \ \sum_{j \in G} u_{ij} x_{ij} \geq c_i + t$$

$$\forall j \in G: \ -\sum_{i \in B} x_{ij} \geq -1$$

$$t \geq 0$$

$$\forall i \in B, \ \forall j \in G: \ x_{ij} \geq 0$$

Again, let y_i's and z_j's be the dual variables corresponding to the first and second set of inequalities, respectively. The dual program is:

$$\text{maximize} \quad \sum_{i \in I} c_i y_i - \sum_{j \in G} z_j \tag{8}$$

$$\text{subject to} \quad \forall i \in B, \ \forall j \in G: \ u_{ij} y_i - z_j \leq 0$$

$$\sum_{i \in B} y_i \leq 1$$

$$\forall i \in B: \ y_i \geq 0$$

$$\forall j \in G: \ z_j \geq 0$$

For this case, let

$$y_i = \frac{1}{\gamma_i n^2} \ \text{ and } \ z_j = \frac{p_j}{n^2}.$$

The first set of inequalities of the dual LP (8) is established by noting that

$$\forall i \in B, \ \forall j \in G: \ \gamma_i \geq \frac{u_{ij}}{p_j}. \ \text{ Hence } \ u_{ij} y_i \leq z_j.$$

Since $\sum_{j \in G} p_j \leq n$, in particular $p_j \leq n$ for each $j \in G$. Furthermore, for each $i \in B$, $j \in G$, $u_{ij} \geq 1$, hence we get that for each $i \in B$, $\gamma_i \geq 1/n$. Hence,

$$\sum_{i \in B} y_i \ = \ \frac{1}{n^2} \sum_{i \in B} \frac{1}{\gamma_i} \leq 1,$$

thereby establishing the second inequality. Hence this solution is feasible for the dual LP (8). Its objective function value is

$$\sum_{i \in B} c_i y_i - \sum_{j \in G} z_j = \frac{1}{n^2} \left(\sum_{i \in B} \alpha_i - \sum_{j \in G} p_j \right) \ = \ \frac{1}{n^2} \sum_{i \in B} \beta_i \ \geq -\frac{1}{\mu},$$

where the second equality follows from Lemma 8. This completes the proof. $\qquad\square$

9 High-Level Description of the Algorithm for ADNB

The full algorithm and its proof appear in Sections 10 and 11, respectively. We will impose the following condition throughout; by Lemma 6, it will ensure that prices are always small.

Invariant: W.r.t. current prices, $\boldsymbol{p}$, $(s, B \cup G \cup t)$ is a min-cut in network $N(\boldsymbol{p})$.

In the Initialization step, we set the money of each buyer to unit to obtain a linear Fisher market and we compute its equilibrium prices using the DPSV algorithm. It is easy to see that these prices satisfy the Invariant for the flexible budget market instance $\mathcal{M}$. Furthermore, these prices satisfy $\sum_{j \in G} p_j = n$. Since Stage I only decreases prices of goods, the condition $\sum_{j \in G} p_j \leq n$ needed in Lemma 9 will be automatically satisfied.

Let f be a balanced flow in $N(\boldsymbol{p})$. Since the Invariant is always maintained, for each buyer i, $\theta_i \geq 0$ and hence $\beta_i \geq -1$. In the algorithm, we will change prices of a well-chosen set J of goods as follows. Multiply the price of each good in J by a variable x and initialize x to 1. In Stage I, we will decrease x and in Stage II we will raise x until the next event happens.

In the next lemma, we will assume that $J = G$ and we will study how the 1-surplus of buyers changes as a function of x. Define $x \cdot f$ to be the flow obtained by multiplying by x the flow on each edge w.r.t. f. Let $\beta_i(x)$ denote i's 1-surplus w.r.t. flow $x \cdot \boldsymbol{p}$. Let B' be the set of buyers having negative 1-surplus w.r.t. prices $\boldsymbol{p}$. If $B' = \emptyset$, define $b = \infty$; else, define $b = \min_{i \in B'} \{-1/\beta_i\}$. Observe that in both cases, $b > 1$.

Lemma 10 *Flow $x \cdot f$ is a balanced flow in $N(x\boldsymbol{p})$ for $0 < x \leq b$, and for each $i \in B$, $\beta_i(x) = x\beta_i$.*

Proof : Since the Invariant holds and f is a max-flow in $N(\boldsymbol{p})$, the cut $(s, G \cup B \cup t)$ is saturated by f, and hence by $x \cdot f$ in $N(x\boldsymbol{p})$. Next we show that $x \cdot f$ is a feasible flow in $N(x\boldsymbol{p})$, i.e., for each buyer $i \in B$, edge (i, t) is not over saturated. Now, $\beta_i = \alpha_i - f(i, t)$. Therefore, the surplus on edge (i, t) w.r.t. flow $x \cdot f$ is $1 + x(\alpha_i - f(i, t)) = 1 + x\beta_i \geq 0$ for $0 < x \leq b$. Hence, edge (i, t) is not over saturated. Furthermore, $\beta_i(x) = x\beta_i$. Finally, since f satisfies Property 1 in $N(\boldsymbol{p})$, $x \cdot f$ satisfies it in $N(x\boldsymbol{p})$, thereby showing that it is a balanced flow. $\square$

Let us give a simplified description of Stage I; for efficiency considerations, the actual algorithm is more complicated. Say that buyer i is *helpful* if $\beta_i < 0$ and *harmful* otherwise. Stage I terminates when either:

- All buyers are rendered helpful, if so, the instance is feasible and the algorithm moves to Stage II to find equilibrium prices.

- $\sum_{i \in B} \beta_i \geq 0$ and $\sum_j p_j > 0$. If so, Lemma 9 yields a proof of infeasibility.

Gradually decrease the prices of goods desired by helpful buyers. As a result, the β's of these buyers increase and, in a sense, we are heading for a proof of infeasibility. However, at some point a new edge may enter the network and on recomputing a balanced flow, some helpful buyers may become harmful or vice versa. Lemma 23 proves that (the more involved

and efficient version of) this iterative process must terminate in one of the 2 conditions given above.

Since Stage II starts with a price vector $\boldsymbol{p}$ that is small, it needs to raise prices of goods to get to the equilibrium. Additionally, at this point buyers have surplus money, and hence Stage II needs to drop their surplus to zero, as demanded by the definition of equilibrium. The feasibility of $\boldsymbol{p}$ ensures that these are compatible goals, and moreover, raising prices does not violate their feasibility. The following lemma establishes this crucial point in the simplified setting of Lemma 10, i.e., prices of all goods are raised; of course, Stage II will raise the prices of well-chosen subsets of G.

Lemma 11 *If in the setting of Lemma 10, prices $\boldsymbol{p}$ are feasible and x is raised without violating the Invariant, then the surplus of each buyer decreases and the resulting price vector is still feasible.*

Proof :   Since $\boldsymbol{p}$ is feasible, for each buyer i, $\beta_i < 0$. Clearly, if $x > 1$, $x \cdot \beta_i < \beta_i$, i.e. the surplus of buyer i decreases. Moreover, the property that the β of each buyer is negative is preserved. Hence the resulting price vector is still feasible. □

10   Details of the Algorithm for ADNB

Algorithm 1 gives the pseudo code for Stage I and Algorithm 2 gives the pseudo code for Stage II[5]. The next two sections give the subroutines used by Stage I and Stage II, respectively, and Section 10.3 gives formal definitions of the predicates used in the While loops.

10.1   Details of Stage I

A run of Stage I is partitioned into *phases*, which are further partitioned into *iterations*. In Stage I, an iteration ends when a new edge is added to the network. A phase ends either when the condition of Step 7 holds or if for some $i \in I, \beta_i \geq 0$. In each iteration, the algorithm computes a balanced flow in the current network, $N(\boldsymbol{p})$.

We employ the following notation. For $J \subseteq G$, define $p(J) = \sum_{j \in J} p_j$ and $\Gamma(J) = \{i \in B \mid \exists j \in J \; s.t. \; (j,i) \in N(\boldsymbol{p})\}$. Similarly, for $I \subseteq B$, define $\alpha(I) = \sum_{i \in I} \alpha_i$, $m(I) = \sum_{i \in I} m_i$ and $\Gamma(I) = \{j \in G \mid \exists i \in B \; s.t. \; (j,i) \in N(\boldsymbol{p})\}$.

The sets B_c and G_c denote the *current sets of buyers and goods* being considered by the algorithm. These sets are initialized to B and G, respectively. At any point the algorithm, $B = B_c \cup B'$ and $G = G_c \cup G'$, where B' and G' are the sets of *adaptable buyers and goods*, respectively; their purpose is explained below. B' and G' are both initialized to $\emptyset$. As the algorithm proceeds, buyers are moved from B_c to B' and goods are moved from G_c to G'.

---

[5]The following power point presentations may make the algorithm easier to understand:
http://www.cc.gatech.edu/ vazirani/Waterloo1.ppt
http://www.cc.gatech.edu/ vazirani/Waterloo2.ppt

As stated in the Introduction, the ostensible goal of Stage I is to arrive at a proof of infeasibility. It does this by decreasing the prices of goods desired by helpful buyers, as a result increasing their β_is and hence increasing $\sum_{i \in B} \beta_i$. In order to make substantial progress in each phase and terminate in polynomial time, Stage I deals with a well-chosen subset, I, of helpful buyers, rather than all helpful buyers. At the start of a phase, the set $I \subseteq B_c$ is initialized to buyers having the smallest β values. The goods they desire are put in set J.

The algorithm lowers the prices of goods in J until a new edge (j, i), with $j \in J$ and $i \in (B_c - I)$ is added to the network. On recomputing a balanced flow, either β_i becomes negative, if so i moves into I and the iteration comes to an end, or for some buyer(s) $i' \in I$, $\beta_{i'}$ increases. If for some buyer $i \in I$, β_i becomes non-negative, the phase comes to an end. Also, if at any point, I and J are found to be adaptable, the algorithm updates B' and G' and the phase comes to an end; in this case also, for the sake of upper bounding the number of phases executed, we will assume that prices of goods in G' are set to zero and for each $i \in B'$, $\beta_i = 0$.

Lemma 23 shows that eventually, either all buyers are rendered helpful or the conditions of Lemma 9 start holding. In the former case, the algorithm moves on to Stage II to find equilibrium prices. One way to view the operation of Stage I is as a tug-of-war between two sets of buyers: the helpful buyers and the harmful buyers. The algorithm decreases the prices of goods desired by buyers in I, thereby increasing their β_is. This helps towards reaching the infeasibility condition stated above. However, as new edges enter the network and a balanced flow is recomputed, buyers may move between the two sets. In particular, if harmful buyers become helpful, then this particular phase actually made some progress towards arriving at a proof of feasibility.

The subroutines used in Stage I are:

- **FindSetsI:** Sets $I \subseteq B_c$ and $J \subseteq G_c$ are initialized as follows.

$$I \leftarrow \arg\min_{i \in B_c}\{\beta_i\} \quad \text{and} \quad J \leftarrow (\Gamma(I) - \Gamma(B_c - I)).$$

Observe that J consists of goods that are the maximum bang-per-buck goods for buyers in I but not for other buyers. All edges from goods in $G_c - J$ to buyers in I are removed; this is justified in Lemma 14 where we show that there is no flow on these edges.

- **UpdateSetsI:** Find the set, I', of all buyers in $B_c - I$ such that there is a residual path, in $R(f) - \{s, t\}$, from a buyer in I to a buyer in I'. Update

$$I \leftarrow (I \cup I');$$

$$J \leftarrow (\Gamma(I) - \Gamma(B_c - I)).$$

All edges from goods in $G_c - J$ to buyers in I are removed. Once again, this is justified in Lemma 14.

Assume that (j, i), $j \in J$, $i \in (B_c - I)$ is the new edge added to S_i in the current iteration; recall that S_i contains i's maximum bang-per-buck goods. Observe that if $I' = \emptyset$, then all the flow from j, which was going to buyers in I before the addition of this edge, must go to i, since

there is no residual path from I to i. Accordingly, **UpdateSetsI** will move good j from J to $G_c - J$. As soon as the prices of goods in J are reduced by an infinitesimally small amount (by decreasing x) buyers in I will not be interested in good j anymore.

Lemma 12 *In Stage I, at the start of each iteration, for each buyer $i \in I$ there is a good $j \in J$ such that edge (j, i) is in the network $N(\boldsymbol{p})$.*

Proof :   Since at the start of each iteration, for each buyer $i \in I$, $\beta_i < 0$, balanced flow must be sending flow on some edge (j, i). Now, by the specification of set J, $j \in J$.   □

We now explain the purpose of the sets B' and G'. Once a good is moved into G', its price is updated in the same way as that of goods in the current set J. As a result, once a buyer enters B' she does not change her preferences. Furthermore, at any point in Stage I, these sets satisfy the following properties:

1. For each buyer $i \in B'$, $\beta_i < 0$.

2. Buyers in B_c are totally uninterested in goods in G' at any price, i.e., for every $i \in B_c$ and every $j \in G'$, $u_{ij} = 0$. Hence, if later the prices of goods in G' are decreased to zero (see below), no edge from B_c to G' will ever enter the network.

The reason for the name "adaptable" is that as far as determining feasibility or infeasibility goes, buyers in B' can be made either helpful or harmful, and hence be made consistent with the outcome of the remaining buyers. If $\forall i \in B_c$, $\beta_i < 0$, we assign goods in G' the prices as computed above hence ensuring that $\forall i \in B'$, $\beta_i < 0$. The resulting price vector is clearly feasible. In Step 9 in Algorithm 1 we have referred to this process as "restoring prices of adjustable goods."

If on the other hand $\sum_{i \in B_c} \beta_i \geq -(n^2/\mu)$, then we set the prices of all goods in G' to zero. Observe that doing this does not introduce any new edges in the network and we get that $\forall i \in B'$, $\beta_i = 0$, thereby ensuring that these buyers don't affect the sum of β_i's. As shown in detail in Section 8.3, all the conditions of Lemma 9 now hold and yield a proof of infeasibility.

Observe that in each iteration, the algorithm needs to compute the largest value of x at which a new edge is added to the network. For any one edge this is straightforward; taking the maximum over all relevant edges gives the required value.

10.2   Details of Stage II

A run of Stage II is also partitioned into *phases*, which are further partitioned into *iterations*. An iteration ends when a new edge is added to the network. A phase ends when a new set goes tight. We will say that $S \subseteq G$ is a *tight set* if the total price of goods in S exactly equals the money possessed by buyers who are interested in goods in S, i.e., $p(S) = m(\Gamma(S))$. Clearly, if S is tight, buyers in $\Gamma(S)$ must have zero surplus and hence have $\beta_i = -1$. In each iteration, the algorithm computes a balanced flow in the current network, $N(\boldsymbol{p})$.

The subroutines used in Stage II are:

- **FindSetsII:** Sets $I \subseteq B$ and $J \subseteq G$ are initialized as follows.

$$I \leftarrow \; \arg\max_{i \in B}\{\theta_i\} \;\; \text{and} \;\; J \leftarrow \; \Gamma(I).$$

All edges are removed from goods in J to buyers in $B - I$; this is justified in Lemma 14 below.

- **UpdateSetsII:** Find the set, I', of all buyers in $B - I$ that have residual paths to buyers in I. Update

$$I \leftarrow (I \cup I');$$

$$J \leftarrow \Gamma(I).$$

All edges are removed from goods in J to buyers in $B - I$. Once again, this is justified in Lemma 14.

Observe that if (j, i) is the new edge added to S_i, the set of i's maximum bang-per-buck goods, then good j must move from $G - J$ to J, whether or not $I' = \emptyset$. The choice of set J above ensures that if the prices of goods in J are increased by an infinitesimally small amount (by increasing x as stated in Algorithm 2), there is no change in the maximum bang-per-buck goods of buyers in $(B - I)$.

Lemma 13 *In Stage II, at the start of each iteration, for each buyer $i \in (B - I)$ there is a good $j \in (G - J)$ such that edge (j, i) is in the network.*

Proof : Since $\theta_i < 1$, the balanced flow must be sending flow on some edge (j, i). If $j \in J$, then there will be a residual path from i to a buyer in I, violating Property 1. Therefore, $j \in (G - J)$. □

Lemma 14 *In Stage I (Stage II), the Invariant holds after all edges from goods in $G - J$ (J) to buyers in I ($B - I$) are removed.*

Proof : The idea of the proof is the same for both statements. In Stage I, right after **UpdateSetsI** is executed, there are no residual paths from I to $B - I$. Therefore, by Property 1, any edges from $G - J$ to I could not be carrying any flow and hence their removal will not affect the Invariant.

In Stage II, right after **UpdateSetsII** is executed, there are no residual paths from $B - I$ to I. Therefore, by Property 1, any edges from J to $B - I$ could not be carrying any flow and hence their removal will not affect the Invariant. □

In each iteration, we need to compute the smallest value of x at which a new edge is added to the network or a new set goes tight. The former computation is the same as in Stage I. Let

the smallest value of x at which a new set goes tight be x^*. Let $b = \min_{i \in I} \left\{ -\frac{1}{\beta_i} \right\}$. Clearly, $b > 1$. Using Lemma 10, proved in Section 8, for x in the range $1 \leq x \leq b$, we prove below that $x^* = b$.

Lemma 15 $x^* = b$.

Proof : By definition of b, for $1 \leq x < b$, for each $i \in I$, the surplus of i will be $1 + x\beta_i > 0$, since $x\beta_i > -1$. Thus, each edge (i, t) will have positive surplus, implying that there are no tight sets.

Next, assume that $x = b$. Let

$$T = \left\{ i \in I \mid -\frac{1}{\beta_i} = b \right\} \quad \text{and} \quad S = \{ j \in J \mid f(j, i) > 0, \text{ for some } i \in T \}.$$

Since f is a balanced flow in $N(\boldsymbol{p})$, there cannot be an edge (j, i) for $j \in S$ and $i \in (I - T)$ in $N(\boldsymbol{p})$. This is so because otherwise there would be a path from T to i in the residual graph, contradicting Property 1 (observe that $\beta_i < -1/b$). Therefore, $\Gamma(S) = T$. Moreover, for $i \in T$, the surplus on edge (i, t) w.r.t. flow $x \cdot f$ in $N(x\boldsymbol{p})$ is $1 + x(-1/b) = 0$. Hence S is a tight set in network $N(x\boldsymbol{p})$ for $x = b$. $\qquad \square$

10.3 Predicates used in While loops

In Step 2 (Stage I), "a proof of feasibility is reached" when $\forall i \in B, \ \beta_i < 0$. "a proof of infeasibility is reached" when

$$\sum_{i \in B} \beta_i \geq -\frac{n^2}{\mu}.$$

In the latter case, Lemma 23 shows that all the conditions of Lemma 9 will be satisfied and the given game is infeasible.

In Step 4 (Stage I), "$B - I$ desire J" is satisfied if and only if $\exists i \in (B - I), \ \exists j \in J : \ u_{ij} > 0$, and "buyers in I have small surplus" is satisfied if and only if $\forall i \in I, \ \beta_i < 0$.

In Step 1 (Stage II), "a buyer in B has surplus money" is satisfied if and only if $\exists i \in B, \ \theta_i > 0$).

In Step 3 (Stage II), "no set in J is tight" is satisfied if and only if $\neg(\exists S, \ \emptyset \subset S \subseteq J \ s.t. \ S$ is tight$)$.

10.4 The role of balanced flow

Besides being used for defining the central notion of feasible prices, balanced flow plays the following three, rather diverse, crucial roles in both stages of our algorithm.

1. Ensure that edges that need to be removed as prices of goods in J are raised did not carry any flow. Hence their removal would not violate the Invariant; this is argued in Lemma 14

2. Ensure that in each iteration, buyers entering I in Stage I (Stage II) have sufficiently large $|\beta_i|$ ($|\theta_i|$); this is established in Lemma 20 (Lemma 29).

3. Prove that sufficient progress is made in an iteration and hence in a phase. This is established in Lemma 21 for Stage I and Lemma 30 for Stage II.

As stated in [DPSV08] (see also Section B), balanced flow could have been defined without resorting to the l_2 norm – as a max-flow that makes the surplus vector lexicographically smallest, after its components are sorted in decreasing order (and hence making the components as balanced as possible). It is easy to prove Property 1 with this definition as well. The first two roles listed above make use of Property 1 only. On the other hand, the third role uses the definition of balanced flow via the l_2 norm and as argued in Section B, the use of the l_2 norm seems indispensable.

Algorithm 1 (Initialization and Stage I of the Algorithm for ADNB)

1. Initialization:

   (i) $\forall i \in B: \quad m_i \leftarrow 1$.

   (ii) Use the DPSV algorithm to compute equilibrium prices, $\boldsymbol{p}$.

   (iii) $\forall i \in B: \quad m_i \leftarrow 1 + \frac{c_i}{\gamma_i}$.

   (iv) $B_c \leftarrow B; \qquad G_c \leftarrow G$.

   (v) $B' \leftarrow \emptyset; \qquad G' \leftarrow \emptyset$.

   (vi) Compute a balanced flow in $N(\boldsymbol{p})$.

<center>

Stage I

</center>

2. *(New Phase)* **While** a proof of feasibility or infeasibility is not reached **do:**

3. **FindSetsI**.

4. *(New Iteration)* **While** $B_c - I$ desire J and buyers in I have small surplus **do:**

5. Multiply the prices of goods in $(J \cup G')$ and α's of buyers in $(I \cup B')$ by x.

   Initialize $x \leftarrow 1$, and decrease x continuously until:

   A new edge (j, i) enters S_i, for $j \in J$ and $i \in (B_c - I)$.
   Add (j, i) to $N(\boldsymbol{p})$ and compute a balanced flow in it.
   UpdateSetsI.

6. **End** *(End Iteration)*

7. If $\forall i \in (B_c - I), \ \forall j \in J: \quad u_{ij} = 0$, then:

   Declare I and J adaptable, i.e,,
   $G' \leftarrow (G' \cup J)$ and $G_c \leftarrow (G_c - J)$.
   $B' \leftarrow (B' \cup I)$ and $B_c \leftarrow (B_c - I)$.

8. **End** *(End Phase)*

9. If $\forall i \in B_c, \ \beta_i < 0$, then:

   Compute a balanced flow in $N(\boldsymbol{p})$.

   Go to Step 1 in Stage II.

10. Else (i.e., $\sum_{i \in B_c} \beta_i \geq 0$), output "The game is infeasible".
    HALT.

```
┌─────────────────────────────────────────────────────────────────────────┐
│  Algorithm  2 (Stage II of the Algorithm for ADNB)                        │
│                                                                           │
│    1. (New Phase) While  a buyer in B has surplus money  do:              │
│                                                                           │
│    2. FindSetsII.                                                         │
│                                                                           │
│    3. (New Iteration) While  no set in J is tight  do:                    │
│                                                                           │
│    4.   Multiply prices of goods in J and α's of buyers in I by x.        │
│         Initialize x ← 1, and raise x continuously until:                 │
│         A new edge (j, i) enters Sᵢ, for j ∈ (G − J) and i ∈ I.           │
│             If so, add (j, i) to N(p) and compute a balanced flow in it.   │
│             UpdateSetsII.                                                  │
│                                                                           │
│    5. End (End Iteration)                                                 │
│                                                                           │
│    6. End (End Phase)                                                     │
│                                                                           │
│    7. Output the current allocations and prices.                          │
│       HALT.                                                               │
└─────────────────────────────────────────────────────────────────────────┘
```

11   Running Time Analysis

We first define some parameters of the given problem instance. Recall that $g = |G|$, $n = |B|$ and $U = \max_{i \in B, j \in G}\{u_{ij}\}$. Let $C = \max_{i \in B} c_i$, and $\Delta = nCU^n$. Observe that program (3) with all $c_i = 0$ is the same as the convex program for a linear Fisher market with all buyers having unit money. Hence, Theorem 2 gives a lower bound of $1/\mu$ on the price of a good computed in Initialization.

The following enhanced version of Lemma 10 will be needed in both stages.

Lemma 16 *Let f be a balanced flow in network $N(\boldsymbol{p})$. Then, for $0 < x \leq b$, the flow $x \cdot f$ is a balanced flow in $N(x\boldsymbol{p})$.*

Proof :   For $i, j \in B$ assume that $1 + x\beta_i < 1 + x\beta_j$. Since $x > 0$, $1 + \beta_i < 1 + \beta_j$, i.e., w.r.t. flow f in $N(\boldsymbol{p})$, the surplus of i is smaller than that of j. Since f is a balanced flow in $N(\boldsymbol{p})$, by Property 1, there is no path from i to j in the residual graph. Therefore, w.r.t. flow $x \cdot f$ in $N(x\boldsymbol{p})$ also there is no path from i to j in the residual graph. Therefore, flow $x \cdot f$ in $N(x\boldsymbol{p})$ satisfies Property 1 and hence is a balanced flow.   □

11.1   Stage I

Throughout Stage I, we will consider a partitioning of B_c into two sets, B_1 and B_2, containing buyers having $\beta_i < 0$ and $\beta_i \geq 0$, respectively. For Stage I, we will work with the following

potential function:

$$\Phi = \sum_{i \in B_1} \beta_i^2.$$

As Stage I proceeds, buyers move from B_c to B', and within B_c between the sets B_1 and B_2. For this reason, it will be convenient to define Φ using an n-dimensional vector, ψ, called the *associated vector* of network N. The i-th component of this vector, ψ_i, is β_i for $i \in B_1$, and is 0 for $i \in (B' \cup B_2)$. Hence, an alternative definition for the potential function is:

$$\Phi = \|\psi\|^2.$$

Lemma 17 *In Stage I, a phase consists of at most ng iterations.*

Proof :   Observe that if in **UpdateSetsI**, $I' = \emptyset$, then a good must move from J to $G_c - J$. Otherwise, a buyer must move from $B_c - I$ to I. Clearly, there can be at most $|J| < |G_c|$ contiguous iterations of the first type and a total of at most $|B_c - I_0| < |B_c|$ iterations of the second type, where I_0 is the set I at the start of the phase. Observe that set I only grows. $\square$

The central fact established below is that Φ drops by a factor of $(1 - 1/(gn^2))$ in a phase (Lemma 22). Towards this end, assume that a given phase consists of k iterations. Let I_0 denote set I at the start of the phase and let I_l denote the set I at the end of the l-th iteration, $1 \leq l \leq k$. Assume that at the start of this phase, $\max_{i \in B_1}\{|\beta_i|\} = \delta = \delta_0$. Clearly, for each $i \in I_0, \beta_i = -\delta_0$. Let

$$\delta_l = \min_{i \in I_l}\{|\beta_i|\}, \quad \text{for} \ \ 1 \leq l < k.$$

As a result of the assumption made in Section 10.1, no matter how the phase ends, $\delta_k = 0$. As we will see in this section, the potential function Φ drops monotonically in each iteration in a phase. Within an iteration, we will account for the drop in two steps. Note that $\beta_i < 0$ for $i \in I$ at the start of a phase. Hence, by Lemma 10, as x decreases, β_i increases. Thus, as prices of goods in J are reduced, the β_i's of buyers $i \in I$ increase, leading to a reduction in Φ. Second, when a new edge (j, i), with $j \in J$ and $i \in (B_c - I)$, is added to the network, the flow becomes more balanced, leading to a further drop. We will account for these two reductions separately, via different arguments (see Lemma 21). For the first step, we work with the l_1 norm, establishing an increase in $\sum_{i \in B_1} \beta_i$. In the second step, $\sum_{i \in B_1} \beta_i$ will not change if $i \in (B_1 - I)$. Instead, we establish a decrease in $\|\psi\|^2$ using an l_2 norm based argument. We observe that the latter argument is difficult to apply to the first step since the money of buyers changes as prices change. Also, we do not know of a simple one step argument that accounts for the entire reduction in an iteration.

Next, we prove a key fact that accounts for the second decrease. Just before new edge (j, i) is added to S_i, let N be the network and $\boldsymbol{p}$ be the prices of goods. Let N' be the network obtained by adding this edge to N; of course, the prices remain unchanged. Let f and f^* be balanced flows in N and N', respectively, and let ψ_i and ψ_i^* be their associated vectors.

Lemma 18 $\|\psi\|^2 - \|\psi^*\|^2 \geq \sum_{h \in B_1} (\psi_h - \psi_h^*)^2$.

Proof :   Since the Invariant holds and the prices are unchanged, f and f^* have the same value. Therefore, flow $f^* - f$ will consist of circulations. Since f is a balanced flow, all these circulations must use the edge (j, i), because otherwise a circulation not using edge (j, i) could be used for making f more balanced. These circulations will have the effect of increasing the surplus of certain buyers in I, say i_l, for $1 \leq l \leq d$, and decreasing the surplus of buyer $i \in (B_c - I)$. Let $\beta_i - \beta_i^* = \nu$, and for $1 \leq l \leq d$, $\beta_{i_l}^* - \beta_{i_l} = \nu_l$. Then, $\sum_{l=1}^{k} \nu_l = \nu$.

For each buyer i_l, $1 \leq l \leq d$, there is a path from i_l to i in the corresponding circulation and hence there is a path from i to i_l in the residual graph w.r.t. flow f^*. Since f^* is balanced, by Property 1, the surplus of buyer i is at least as large as that of i_l. Therefore, $\beta_i^* \geq \beta_{i_l}^*$.

In going from N to N', the ψ_h values can change only for $h = i$, and $h = i_l$, for $1 \leq l \leq d$. We will consider 3 cases.

Case 1: $\beta_i \geq 0$, i.e., $i \in B_2$, and $\beta_i^* \geq 0$.
In this case, $\psi_i = \psi_i^* = 0$ and the lemma is obvious.

Case 2: $\beta_i \geq 0$, i.e., $i \in B_2$, and $\beta_i^* < 0$.
Let $a = -\beta_i^*$. In this case, $\psi_i = 0$ and $\psi_i^* = -a$. Also, let $b_l = -\beta_{i_l}$, for $1 \leq l \leq d$.

Clearly, $a \leq \sum_{l=1}^{k} \nu_l$. Since $\beta_i^* \geq \beta_{i_l}^*$, $a \leq b_l - \nu_l$. Now,

$$\|\psi\|^2 - \|\psi^*\|^2 = \left(0^2 + \sum_{l=1}^{k} b_l^2 \right) - \left(a^2 + \sum_{l=1}^{k} (b_l - \nu_l)^2 \right) = -a^2 + \sum_{l=1}^{k} (2b_l - \nu_l)\nu_l$$

$$\geq -a^2 + \sum_{l=1}^{k} (2a + \nu_l)\nu_l \geq -a^2 + \sum_{l=1}^{k} \nu_l^2 + 2a \sum_{l=1}^{k} \nu_l \geq a^2 + \sum_{l=1}^{k} \nu_l^2,$$

where the first inequality follows from $a \leq b_l - \nu_l$ and the third one from $a \leq \sum_{l=1}^{k} \nu_l$.

Case 3: $\beta_i^* < 0$, i.e., $i \in (B_1 - I)$.
Clearly, in this case, $\beta_i < 0$. Let $a = -\beta_i$, and for $1 \leq i_l \leq d$, let $b_l = -\beta_{i_l}$. Now, apply Lemma 19, to get $\|\psi\|^2 - \|\psi^*\|^2 \geq \nu^2$. Clearly, $\nu^2 \geq \sum_{l=1}^{k} \delta_l^2$, giving the lemma.   □

Lemma 19 *Let $\delta, \delta_l \geq 0$, $l = 1, 2, \ldots, k$, with $\delta = \sum_{l=1}^{k} \delta_l$. If $a + \delta \leq b_l - \delta_l$, for $l = 1, 2, \ldots, k$ then*

$$\|(a, b_1, b_2, \ldots, b_k)\|^2 - \|(a + \delta, b_1 - \delta_1, b_2 - \delta_2, \ldots, b_k - \delta_k)\|^2 \geq \delta^2.$$

Proof :

$$\left(a^2 + \sum_{l=1}^{k} b_l^2 \right) - \left((a + \delta)^2 + \sum_{l=1}^{k} (b_l - \delta_l)^2 \right)$$

$$\geq \left((a + \delta - \delta)^2 + \sum_{l=1}^{k}(b_l - \delta_l + \delta_l)^2\right) - \left((a + \delta)^2 + \sum_{l=1}^{k}(b_l - \delta_l)^2\right)$$

$$\geq \delta^2 + 2(a + \delta)\left((\sum_{l=1}^{k}\delta_l) - \delta\right) \quad \geq \quad \delta^2.$$

$\square$

Let ψ^0 denote vector ψ at the start of the phase and ψ^l denote ψ at the end of iteration l, for $1 \leq l \leq k$.

Lemma 20 *In the l-th iteration, there is a buyer $i \in I_{l-1}$ such that $|\beta_i|$ decreases by at least $(\delta_{l-1} - \delta_l)$, for $1 \leq l < k$.*

Proof : By Property 1, if there is a residual path from i to i', then $\theta_i \geq \theta_{i'}$. Hence, by the definition of set I' in procedure **UpdateSetsI**, there is a buyer $i \in I_{l-1}$ which achieves $\min_{i \in I_l}\{|\beta_i|\}$ at the end of iteration l. Clearly, β_i increases (and hence $|\beta_i|$ decreases) by at least $(\delta_{l-1} - \delta_l)$ in the l-th iteration, for $1 \leq l < k$. $\square$

Lemma 21 *For $1 \leq l \leq k$,*

$$\|\psi^{l-1}\|^2 - \|\psi^l\|^2 \geq (\delta_{l-1} - \delta_l)^2.$$

Proof : We first prove the statement for $1 \leq l < k$. By Lemma 20, there is a buyer $i \in I_{l-1}$ such that β_i increases by at least $(\delta_{l-1} - \delta_l)$ in the l-th iteration. Let us split this increase into two parts, the increase due to decrease in the prices of goods in J and that due to a new edge entering the network. Let these be a and b, respectively. Therefore, $a + b = \delta_{l-1} - \delta_l$.

Let ψ' be the vector ψ just before the new edge is added to the network in iteration l, i.e., right after all the decrease in prices of J has happened. As prices in J decrease, the βs of buyers in I increase, each leading to a decrease in $\|\psi'\|^2$; clearly, the βs of buyers in $B_c - I$ remain unchanged. Let c be the value of β_i at the beginning of iteration l. Clearly, $-c > a + b$. Now,

$$\|\psi^{l-1}\|^2 - \|\psi'\|^2 \geq c^2 - (c+a)^2 = -a^2 - 2ac.$$

By Lemma 18,

$$\|\psi'\|^2 - \|\psi^l\|^2 \geq b^2.$$

Adding the two and using $-c > a + b$ we get

$$\|\psi^{l-1}\|^2 - \|\psi^l\|^2 \geq -a^2 - 2ac + b^2 > a^2 + 2ab + b^2 = (a+b)^2 \geq (\delta_{l-1} - \delta_l)^2,$$

where the second last inequality follows from the observation that $b \leq -c$.

31

Finally, in the k-th iteration, there is a buyer $i \in I_{k-1}$ whose ψ_i changes from $\beta_i < 0$ to 0. Therefore,

$$\|\psi^{k-1}\|^2 - \|\psi^k\|^2 \geq \beta_i^2 \geq (\delta_{k-1} - \delta_k)^2,$$

since $\delta_{k-1} \leq -\beta_i$ and $\delta_k = 0$. □

Lemma 22 *In a phase in Stage I, the potential drops by a factor of*

$$\left(1 - \frac{1}{n^2 g}\right).$$

Proof :   Now, $\|\psi^0\|^2 - \|\psi^k\|^2$ can be written as a telescoping sum of k terms, each of which is the decrease in the potential in one of the k iterations. Lemma 21 gives a lower bound on each of these terms. The total lower bound is minimized when each of the differences $(\delta_{l-1} - \delta_l)$ is equal. Now using the fact that $\delta_0 = \delta$ and $\delta_k = 0$, we get:

$$\|\psi^0\|^2 - \|\psi^k\|^2 \geq \frac{\delta^2}{k}.$$

Finally, since $\|\psi^0\|^2 \leq n\delta^2$, and by Lemma 17 $k \leq ng$, we get:

$$\|\psi^k\|^2 \leq \|\psi^0\|^2 \left(1 - \frac{1}{n^2 g}\right).$$

□

Lemma 23 *Stage I terminates with either a feasible price vector or a proof of infeasibility. Moreover, its execution requires at most*

$$O\left(n^5 g^3 \log U\right)$$

max-flow computations.

Proof :   At the start of the algorithm, the potential is at most n. Assume that the feasibility condition is not reached in the execution of Stage I. Then, as soon as the potential drops below $n^3/(\mu^2)$,

$$\sum_{i \in B_1} |\beta_i| \leq \frac{n^2}{\mu}.$$

Hence, at this point, $\sum_{i \in B} \beta_i \geq -(n^2/\mu)$, therefore, the algorithm will halt. We next show that in this case, all the conditions of Lemma 9 hold and market $\mathcal{M}$ is indeed infeasible.

As stated in Section 9, the initialization and the fact that Stage I only decreases prices of goods will ensure that the condition $\sum_{j \in G} p_j \leq n$ is satisfied. Also, since G_c must be non-empty, there is a good having positive price and hence $\sum_{j \in G} p_j > 0$.

By Lemma 22, the potential drops by a factor of two after $O(n^2 g)$ type 1 phases. Hence, an upper bound on the number of such phases needed is

$$O\left(n^2 g \log\left(\frac{n\mu^2}{n^3}\right)\right) \;=\; O(n^3 g^2 \log U).$$

Clearly, this dominates the number of type 2 phases.

By Lemma 17 each phase consists of at most ng iterations and each iteration requires n max-flow computations for finding a balanced flow. The lemma follows. $\qquad\square$

11.2 Stage II

When $\forall i \in B : \; \beta_i < 0$, the algorithm starts with Stage II. Since in this stage the algorithm only raises prices of goods (i.e., increases x), by Lemma 11, $\forall i \in B : \; \beta_i < 0$ holds until termination.

In this section, we will work with the θ_is of buyers, rather than their β_is. Thus, throughout Stage II, $\forall i \in B : \; \theta_i < 1$. For Stage II, we will use the following potential function:

$$\Phi = \sum_{i \in B} \theta_i^2.$$

Lemma 24 *In Stage II, at the termination of a phase, the prices of goods in the newly tight set must be rational numbers with denominator $\leq \Delta$.*

Proof :   The idea is to show that prices can be obtained by solving a system of linear equations. Let S be the newly tight set. Consider the subgraph of the network induced on the bipartition $(S, \Gamma(S))$, and view this as an undirected graph, say H. Assume w.l.o.g. that this graph is connected (otherwise we prove the lemma for each connected component of H). Pick a spanning tree in H.

Pick any good $j \in S$, and find a path in the spanning tree from j to each good $j' \in S$. If j reaches j' with a path of length $2l$, then $p_{j'} = ap_j/b$ where a and b are products of l utility parameters (u_{ik}'s) each. Since alternate edges of this path contribute to a and b, we can partition the u_{ik}'s of edges in the spanning tree into two sets, T_1 and T_2, such that a uses u_{ik}'s from T_1 and b uses those from T_2.

Next, consider $\alpha_i = c_i/\gamma_i$, for $i \in \Gamma(S)$. Now, $\gamma_i = u_{i,j'}/p_{j'}$, where (i, j') is any edge in the network. Find the path in the spanning tree from i to j and use the first edge on this path for computing γ_i (it is easy to see that this edge will come from T_2), and substitute $p_{j'}$ using the expression stated above, i.e., $p_{j'} = ap_j/b$.

Since S is a tight set,

$$\sum_{j' \in S} p_{j'} = \sum_{i \in \Gamma(S)} 1 + \alpha_i.$$

In this equation, substitute for $p_{j'}$ and α_i using the expressions constructed above to get an equation with one variable, i.e., p_j. Now, the denominator of p_j will be the product of $|T_2|$ terms each of size at most U. Hence, the denominator of p_j is $\leq \Delta$. $\qquad\square$

Lemma 25 *In Stage II, consider two phases P and P', not necessarily consecutive, with P' subsequent to P and such that good j lies in the newly tight sets at the end of P as well as P'. Then the increase in the price of j, going from P to P', is at least $1/\Delta^2$.*

Proof :   Let the prices of j at the end of P and P' be p/q and r/s, respectively. Since prices only increase in Stage II and during a phase, prices in the newly tight set must increase, $r/s > p/q$. By Lemma 24, $q \leq \Delta$ and $r \leq \Delta$. Therefore the increase in the price of j,

$$\frac{r}{s} - \frac{p}{q} \geq \frac{1}{\Delta^2}.$$

$\qquad\square$

Lemma 26 *In Stage II, a phase consists of at most g iterations.*

Proof :   After each iteration, other than the last one, at least one good will move from $G - J$ to J. Moreover, the algorithm never moves a good from J to $G - J$ during the phase. Hence, the lemma follows. $\qquad\square$

The structure of the rest of the argument is quite similar to that of Stage I. Once again, the central fact established is that Φ drops by an inverse polynomial factor, of $(1 - 1/ng)$, in a phase (Lemma 31). Assume that a given phase consists of k iterations. Let I_0 denote set I at the start of the phase, and let I_l denote the set I at the end of the l-th iteration, $1 \leq l \leq k$. Assume that at the start of this phase, $\max_{i \in B}\{\theta_i\} = \delta = \delta_0$. Let

$$\delta_l = \min_{i \in I_l}\{\theta_i\}, \quad \text{for} \quad 1 \leq l < k, \quad \text{and} \quad \theta_k = 0.$$

As in Stage I, we will account for the drop in Φ in two steps in each iteration. First, as prices of goods in J are increased, the θ_i's of buyers $i \in I$ decrease, leading to a reduction in Φ. Second, when a new edge (j, i), with $j \in (G_c - J)$ and $i \in I$, is added to the network, the flow becomes more balanced, leading to a further drop. As in Stage I, we will account for the first drop using the l_1 norm and the second drop using the l_2 norm.

We begin by accounting for the second decrease. Just before new edge (j, i) is added, let N be the network and $\boldsymbol{p}$ be the prices of goods. Let N' be the network obtained by adding this

edge to N; of course, the prices remain unchanged. Let f and f^* be balanced flows in N and N', respectively. Denote by θ (θ^*) the surplus vector w.r.t. flow f in N (flow f^* in N').

Lemma 27 $\|\theta\|^2 - \|\theta^*\|^2 \geq \nu^2$, where $\nu = \theta_i - \theta_i^*$.

Proof : Since the Invariant holds and the prices are unchanged, f and f^* have the same value. Therefore, flow $f^* - f$ will consist of circulations. Since f is a balanced flow, all these circulations must use the edge (j, i), because otherwise a circulation not using edge (j, i) could be used for making f more balanced. These circulations will have the effect of decreasing the surplus of buyer $i \in I$, and increasing the surplus of buyers $i_l \in (B - I)$, for $1 \leq l \leq d$. Let $\theta_{i_l}^* - \theta_{i_l} = \nu_l$, for $1 \leq l \leq d$. Then, $\sum_{l=1}^{h} \nu_l = \nu$.

For each buyer i_l, $1 \leq i_l \leq d$, there is a path from i_l to i in the corresponding circulation and hence there is a path from i to i_l in the residual graph w.r.t. flow f^*. Since f^* is balanced, by Property 1, the surplus of buyer i w.r.t. f^* is at least as large as that of i_l. Therefore, $\theta_i^* \geq \theta_{i_l}^*$. The inequality $\|\theta\|^2 - \|\theta^*\|^2 \geq \nu^2$ now follows from Lemma 28, on substituting $a = \theta_i^*$, and for $1 \leq l \leq d$, $b_l = \theta_{i_l}^*$. $\qquad \square$

Lemma 28 If $a \geq b_l \geq 0, l = 1, 2, \ldots, k$ and $\delta = \sum_{l=1}^{k} \delta_l$ where $\delta, \delta_l \geq 0, l = 1, 2, \ldots, k$, then

$$\|(a + \delta, b_1 - \delta_1, b_2 - \delta_2, \ldots, b_k - \delta_k)\|^2 - \|(a, b_1, b_2, \ldots, b_k)\|^2 \geq \delta^2.$$

Proof :
$$(a + \delta)^2 + \sum_{i=1}^{k} (b_i - \delta_i)^2 - a^2 - \sum_{i=1}^{k} b_i^2 \geq \delta^2 + 2a(\delta - \sum_{i=1}^{k} \delta_i) \geq \delta^2.$$
$\qquad \square$

Let θ^0 denote the surplus vector at the start of the phase and let θ^l denote the surplus vector at the end of iteration l, for $1 \leq l \leq k$.

Lemma 29 In the l-th iteration, there is a buyer $i \in I_{l-1}$ whose surplus decreases by at least $(\delta_{l-1} - \delta_l)$, for $1 \leq l < k$.

Proof : By Property 1, if there is a residual path from i to i', then $\theta_i \geq \theta_{i'}$. Hence, by the definition of set I' in procedure **UpdateSetsII**, there is a buyer $i \in I_{l-1}$ which achieves $\min_{i \in I_l}\{|\theta_i|\}$ at the end of iteration l. Clearly, the surplus of i decreases by at least $(\delta_{l-1} - \delta_l)$ in the l-th iteration, for $1 \leq l < k$. $\qquad \square$

Lemma 30 For $1 \leq l \leq k$,

$$\|\theta^{l-1}\|^2 - \|\theta^l\|^2 \geq (\delta_{l-1} - \delta_l)^2.$$

Proof :   We first prove the statement for $1 \leq l < k$. By Lemma 29, there is a buyer $i \in I_{l-1}$ whose surplus decreases by at least $(\delta_{l-1} - \delta_l)$ in the l-th iteration. Let us split this decrease into two parts, the decrease due to increase in the prices of goods in J and that due to a new edge entering the network. Let these be a and b, respectively. Clearly, $a + b \geq \delta_{l-1} - \delta_l$.

Let θ' be the surplus vector just before the new edge is added to the network in iteration l, i.e., right after all the increase in prices of J has happened. As prices in J increase, the surpluses of buyers in I decrease, but those of buyers in $B - I$ remain unchanged. Let c be the surplus of buyer i at the beginning of iteration l. Clearly, $c > a + b$. Now,

$$\|\theta^{l-1}\|^2 - \|\theta'\|^2 \geq c^2 - (c-a)^2 = -a^2 + 2ac.$$

By Lemma 27,

$$\|\theta'\|^2 - \|\theta^l\|^2 \geq b^2.$$

Adding the two we get

$$\|\theta^{l-1}\|^2 - \|\theta^l\|^2 \geq -a^2 + 2ac + b^2 > a^2 + 2ab + b^2 = (a+b)^2 \geq (\delta_{l-1} - \delta_l)^2,$$

where the second to last inequality follows from the observation that $b \leq c$.

Finally, in the k-th iteration, there is a buyer $i \in I_{k-1}$ whose surplus changes from $\theta_i > 0$ to 0. Therefore,

$$\|\theta^{k-1}\|^2 - \|\theta^k\|^2 \geq \theta_i^2 \geq (\delta_{k-1} - \delta_k)^2,$$

since $\delta_{k-1} \leq \theta_i$ and $\delta_k = 0$. $\qquad\qquad\square$

Lemma 31 *In a phase in Stage II, the potential drops by a factor of*

$$\left(1 - \frac{1}{ng}\right).$$

Proof :   Now, $\|\theta^0\|^2 - \|\theta^k\|^2$ can be written as a telescoping sum of k terms, each of which is the decrease in the potential in one of the k iterations. Lemma 30 gives a lower bound on each of these terms. The total lower bound is minimized when each of the differences $(\delta_{l-1} - \delta_l)$ is equal. Now using the fact that $\delta_0 = \delta$ and $\delta_k = 0$, we get:

$$\|\theta^0\|^2 - \|\theta^k\|^2 \geq \frac{\delta^2}{k}.$$

Finally, since $\|\theta^0\|^2 \leq n\delta^2$, and by Lemma 26 $k \leq g$, we get:

$$\|\theta^k\|^2 \leq \|\theta^0\|^2 \left(1 - \frac{1}{n^2}\right).$$

$\qquad\qquad\square$

Lemma 32 *The execution of Stage II requires at most*

$$O\left(n^2 g^2 (n \log U + \log C)\right)$$

max-flow computations.

Proof : By Lemma 31, the potential drops by a factor of half after $O(ng)$ phases. At the start of Stage II, the potential is at most n. Once its value drops below $1/\Delta^4$, by Lemma 24 and Lemma 25 in at most n more phases, the price of some good must increase by at least $1/\Delta^2$, leading to a corresponding decrease in the surplus money. Hence, the potential must drop by $1/\Delta^4$ hence obtaining equilibrium prices. Therefore the number of phases is

$$O(ng \log(\Delta^4 n)) \;=\; O(ng(n \log U + \log C)).$$

By Lemma 26 each phase consists of g iterations and each iteration requires n max-flow computations for computing a balanced flow. The lemma follows. $\square$

Lemmas 23 and 32 give:

Theorem 33 *Algorithms 1 and 2 solve the decision and search versions of Nash bargaining game* **ADNB** *using*

$$O\left(n^2 g^2 (n^3 g \log U + \log C)\right)$$

max-flow computations.

12 How Were the KKT Conditions Relaxed?

As pointed out in Section 4 in [DPSV08], the primal-dual paradigm operates in a fundamentally different way in the setting of logarithmic RCPs than in the setting of an integral linear program. In the latter setting, in each iteration, it picks an unsatisfied complementary slackness condition and satisfies it. On the other hand, in the former setting, the algorithm starts off with a suboptimal solution that can be viewed as relaxing a class of the KKT conditions. It then tightens these conditions gradually; when they are all fully tightened, the optimal solution has been reached.

Let us first show that this high level picture applies to Stage II of our algorithm as well. Consider the situation right after a balanced flow has been computed at any point in Stage II, and consider an arbitrary buyer i. At this point, let f_i be the flow sent on edge (i, t) by the balanced flow; f_i is also the money spent by i in the current allocation. Buyer i's available money at this point is $m_i = 1 + c_i/\gamma_i$. Therefore,

$$f_i = m_i - \beta_i - 1 = \frac{c_i}{\gamma_i} - \beta_i.$$

Let v_i be the total utility derived by i from the current allocation and suppose that $x_{ij} > 0$. Then,

$$\gamma_i = \frac{u_{ij}}{p_j} = \frac{v_i}{f_i} = \frac{v_i}{(-\beta_i) + \frac{c_i}{\gamma_i}}.$$

This yields

$$\left((-\beta_i) + \frac{c_i}{\gamma_i}\right)\gamma_i = v_i \Rightarrow \gamma_i = \frac{v_i - c_i}{-\beta_i}.$$

Substituting for γ_i and rearranging we get

$$\frac{p_j}{-\beta_i} = \frac{u_{ij}}{v_i - c_i}.$$

To summarize, at any point in Stage II, we have ensured the first two KKT conditions and relaxed the last two as follows:

(1) $\forall j \in G: \ p_j \geq 0$.

(2) $\forall j \in G: \ p_j > 0 \Rightarrow \sum_{i \in B} x_{ij} = 1$.

(3') $\forall i \in B, \ \forall j \in G: \ \frac{p_j}{-\beta_i} \geq \frac{u_{ij}}{v_i - c_i}$.

(4') $\forall i \in B, \ \forall j \in G: x_{ij} > 0 \Rightarrow \frac{p_j}{-\beta_i} = \frac{u_{ij}}{v_i - c_i}$.

Observe that if prices are not feasible, then for some i, $\beta_i \geq 0$. If so, the relaxed KKT conditions (3') and (4') will be meaningless. Recall that throughout Stage II, $0 < -\beta_i \leq 1$ and $(-\beta_i)$ monotonically increases and reaches 1 at termination. Thus, at termination, the last two KKT conditions are also ensured. For establishing a bound on the number of phases needed in Stage II, it suffices to study the potential function

$$\Phi' = \sum_{i \in B} \beta_i^2.$$

Clearly, $\Phi' > 0$ at the start of Stage II and increases monotonically to n. However, it turns out to be more convenient to study the potential function Φ given in Section 11.2, which clearly achieves the same end.

13   Discussion

Perhaps the most basic question remaining about RCPs is whether there are other classes of such programs besides the two given in this paper. It is quite clear that the answer should eventually involve the use of algebraic geometry.

Sturmfels and Uhler [SU10] have given the following interesting RCP which does not lie in either of the classes of RCPs defined in this paper. Let S be an $n \times n$ positive semidefinite

matrix; this is the sample covariance matrix. Let $G = (V, E)$ be a graph on n vertices. They prove that if G is chordal, then the following is an RCP:

$$\text{minimize} \quad \log \det \Sigma \tag{9}$$
$$\text{subject to} \quad \forall (i, j) \ s.t. \ (i, j) \in E \ \text{or} \ i = j : \quad \Sigma_{ij} = S_{ij}$$

We ask if one can define a class of RCPs that contains this and other RCPs.

As in the case of the EG-program, the optimal solutions of convex program (3) resemble those of a linear program rather than a nonlinear program. So, we repeat a question raised in [Vaz07] namely, can the solution to **ADNB** be captured via a linear program? We believe the answers to these questions are "no" and that establishing this in a suitable formal framework will provide new insights into the boundary between linear and nonlinear programs.

For the linear case of the Arrow-Debreu market model Jain gives a rather unusual convex program [Jai07] which yields a polynomial time algorithm. Eaves [Eav76] gives an LCP for this model, hence showing that it always has a rational equilibrium if all parameters are rational. However, rationality cannot be established directly from Jain's program and no combinatorial algorithm is known. We believe that there should be an RCP for this market model that directly yields a proof of rationality. Also, in the absence of a combinatorial algorithm, we ask if there is a polynomial time algorithm for this market that uses an LP solver, i.e., under Program B? A much more general question along these lines is whether all RCPs have a strongly polynomial algorithm under Program B. Perhaps the latter question can be made easier by assuming knowledge of a proof of rationality via an exponential family of LPs, one of which yields the optimal solution to the RCP. Another question, stated in the Introduction, is to obtain a strongly polynomial algorithm for solving quadratic RCPs, assuming an LP-oracle.

Next, we list some RCPs and leave the problem of finding combinatorial algorithms for them. First, generalize **ADNB** to additively separable, piecewise-linear, concave utilities. This problem has a rational convex program, and the question of finding a combinatorial algorithm for it becomes even more significant in view of recent results showing PPAD-completeness of computing an equilibrium in the Arrow-Debreu model with these utility functions [CDDT09, CT09, VY11].

In an interesting paper, Kalai [Kal77] relaxed Nash's axiom of symmetry and derived the solution concept of *nonsymmetric bargaining games*. The convex program capturing the solution to the nonsymmetric extension of **ADNB** is also rational; moreover, this convex program generalizes the Eisenberg-Gale program and hence captures Fisher's linear case as well. Despite substantial effort, this problem has not yielded to a combinatorial algorithm. Once it is obtained, one could consider the common generalization of the last two problems, i.e., nonsymmetric **ADNB** with additively separable, piecewise-linear, concave utilities.

On restricting **ADNB** (nonsymmetric **ADNB**) to zero disagreement utilities we get the problems of computing equilibria for linear Fisher markets with unit (arbitrary) money among buyers. Of course, both these problems are total. It turns out that a combinatorial algorithm for the unit money case is no easier than that for the arbitrary money case. In view of this, the difficulty of obtaining a combinatorial algorithm for nonsymmetric **ADNB** comes as a surprise and may be substantiating the observation that in the setting of logarithmic RCPs, non-total

problems behave quite differently from total problems.

It is easy to see that if the given market is feasible and Stage I is started off with *any* small price vector, not necessarily the one found by Initialization, it will terminate with a feasible price vector. Hence we get the following interesting fact:

Lemma 34 *Let $\mathcal{M}$ be a feasible flexible budget market and let $\boldsymbol{p}$ be small, positive prices for it. Then, there exist positive prices $\boldsymbol{q}$ such that $\boldsymbol{p}$ weakly dominates $\boldsymbol{q}$ and prices $\boldsymbol{q}$ are feasible.*

In view of Lemma 6 and the remarks made after it, and of Lemma 34, characterizing the sets of small and feasible price vectors for a feasible flexible budget market is an interesting question.

Approximation algorithms came as a natural generalization of combinatorial optimization, which studied exact algorithms. In the same vein, we believe that combinatorial algorithms for RCPs should naturally lead to the study and design of combinatorial algorithms that yield rational approximations to special families of nonlinear convex programs.

References

[And09]   M. Andrews. Personal communication. 2009.

[AQS05]   M. Andrews, L. Qian, and A. Stolyar. Optimal utility based multi-user throughput allocation subject to throughput constraints. In *INFOCOM*, 2005.

[BDX10]   B. Birnbaum, N. Devanur, and L. Xiao. New convex programs and distributed algorithms for Fisher markets with linear and spending constraint utilities. Unpublished manuscript, 2010.

[BS00]   W. C. Brainard and H. E. Scarf. How to compute equilibrium prices in 1891. *Cowles Foundation Discussion Paper*, (1270), 2000.

[BTN01]   A. Ben-Tal and A. Nemirovski. On polyhedral approximations of the second order cone. *Mathematics of Operations Research*, 26(2):193–205, 2001.

[BV04]   S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[CDDT09]   X. Chen, D. Dai, Y. Du, and S.-H. Teng. Settling the complexity of arrow-debreu equilibria in markets with additively separable utilities. In *IEEE Annual Symposium on Foundations of Computer Science*, 2009.

[CDV10]   D. Chakrabarty, N. Devanur, and V. V. Vazirani. New results on rationality and strongly polynomial solvability in Eisenberg-Gale markets. *SIAM Journal on Discrete Mathematics*, 24(3), 2010.

[CT09]   X. Chen and S.-H. Teng. Spending is not easier than trading: on the computational equivalence of Fisher and Arrow-Debreu equilibria. In *ISAAC: International Symposium on Algorithms and Complexity*, pages 647–656, 2009.

[DPSV08]   N. Devanur, C. H. Papadimitriou, A. Saberi, and V. V. Vazirani. Market equilibrium via a primal-dual-type algorithm. *Journal of the ACM*, 55(5), 2008.

[Eav76]   B. C. Eaves. A finite algorithm for the linear exchange model. *Journal of Mathematical Economics*, 3(2):197–203, 1976.

[Edm65]   J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[EG59]   E. Eisenberg and D. Gale. Consensus of subjective probabilities: the Pari-Mutuel method. *The Annals of Mathematical Statistics*, 30:165–168, 1959.

[FK92]   A. Frank and A. Karzanov. Determining the distance to the perfect matching polytope of a bipartite graph. Unpublished manuscript, 1992.

[GLS88]   M. Grotschel, L. Lovasz, and A. Schirjver. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag, 1988.

[GV11]   G. Goel and V. V. Vazirani. A perfect price discrimination market model with production, and a rational convex program for it. *Mathematics of Operations Research*, 36, 2011.

[HKL80]   R. Helgason, J. Kennington, and H. Lall. A polynomially bounded algorithm for a singly constrained quadratic program. *Mathematical Programming*, 18:338–343, 1980.

[HS90]   D. Hochbaum and J. G. Shantikumar. Convex separable optimization is not much harder than linear optimization. *Journal of the ACM*, 37(4), 1990.

[Jai07]   K. Jain. A polynomial time algorithm for computing the Arrow-Debreu market equilibrium for linear utilities. *SIACOMP*, 37(1):303–318, 2007.

[JPP00]   A. Jalali, R. Padovani, and R. Pankaj. Data throughput of CDMA-HDR a high efficiency-high data rate personal communication wireless system. In *Proceedings of the IEEE Semiannual Vehicular Technology Conference, VTC2000-Spring*, Tokyo, Japan, May 2000.

[JV08]   K. Jain and V. V. Vazirani. Eisenberg-Gale markets: Algorithms and game-theoretic properties. *Games and Economic Behavior*, 70(1), 2008.

[Kal77]   E. Kalai. Nonsymmetric Nash solutions and replications of 2-person bargaining. *International Journal of Game Theory*, 6(3):129–133, 1977.

[Kal85]   E. Kalai. Solutions to the bargaining problem. In L. Hurwicz, D. Schmeidler, and H. Sonnenschein, editors, *Social Goals and Social Organization*, pages 75–105. Cambridge University Press, 1985.

[Kel97]   F. P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[KM97]   A. V. Karzanov and S. T. McCormick. Polynomial methods for separable convex optimization in unimodular linear spaces with applications. *SIAM Journal on Computing*, 26(4):1245–1276, 1997.

[LFB06]   M. Lobo, M. Fazel, and S. Boyd. Portfolio optimization with linear and fixed costs. *Ann.Oper. Res.*, 2006.

[Min84]   M. Minoux. A polynomial algorithm for minimum quadratic cost flow problems. *European J. of OR*, 18(3):377–387, 1984.

[Mit]   Hans Mittlemann. Benchmarks for optimization software. Url: http://plato.asu.edu/bench.html.

[Nas50]   J. F. Nash. The bargaining problem. *Econometrica*, 18:155–162, 1950.

[NBC⁺09] N. Nisan, J. Bayer, D. Chandra, T. Franji, R. Gardner, Y. Matias, N. Rhodes, M. Seltzer, D. Tom, and H. Varian. Google's auction for TV ads. In *International Colloquium on Automata, Languages, and Programming*, 2009.

[Nis09]   N. Nisan. Report from ALGO 2009. In *Algorithmic Game Theory Blog*, September 10, 2009.

[OR94]   M. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.

[Orl10]   J. B. Orlin. Improved algorithms for computing Fisher's market clearing prices. In *ACM Symposium on the Theory of Computing*, 2010.

[Sch03]   A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80:346–355, 2003.

[SU10]   B. Sturmfels and C. Uhler. Multivariate Gaussians, semidefinite matrix completion, and convex algebraic geometry. *Annals of the Institute of Statistical Mathematics*, 62:603–638, 2010.

[TB00]   P. Tseng and D. Bertsekas. An ϵ-relaxation method for separable convex cost generalized network flow problems. *Mathematical Programming*, 88:85–104, 2000.

[TL89]   W. Thomson and T. Lensberg. *Axiomatic Theory of Bargaining With a Variable Population*. Cambridge University Press, 1989.

[Tse]   D. Tse. Multiuser diversity in wireless networks. http://www.eecs.berkeley.edu/~dtse/stanford416.ps.

[Vaz]   V. V. Vazirani. Rational convex programs and efficient algorithms for 2-player Nash and nonsymmetric bargaining games. *SIAM Journal on Discrete Mathematics*. To appear.

[Vaz07]   V. V. Vazirani. Combinatorial algorithms for market equilibria. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, pages 103–134. Cambridge University Press, 2007.

[Vaz10]   V. V. Vazirani. Spending constraint utilities, with applications to the Adwords market. *Mathematics of Operations Research*, 35(2), 2010.

[VY11]   V. V. Vazirani and M. Yannakakis. Market equilibria under separable, piecewise-linear, concave utilities. *Journal of the ACM*, 58(3), 2011.

14   Acknowledgments

A   Solution to ADNB in the Limit

Assume that we are given an instance of game **ADNB** that is feasible and let $\mathcal{M}$ be the flexible budget market obtained from it. In this section, we present an algorithm that converges to the equilibrium of $\mathcal{M}$ in the limit.

Algorithm 3 will use the DPSV algorithm as a subroutine [DPSV08]. When this subroutine is called, we assume that the money of each agent is fixed and is specified in the vector $\boldsymbol{m}$.

Let $N'(\boldsymbol{p})$ denote the network for the case that the money of agents is fixed and specified by vector $\boldsymbol{m}$; this network differs from network $N(\boldsymbol{p})$ only in that the capacities of edges going from buyers to t are specified by $\boldsymbol{m}$, rather than being defined as a function of the prices.

Algorithm 3 (Solution to ADNB in the Limit)

1. Initialization: $\forall i \in B : \quad m_i \leftarrow 1$.

2. Compute equilibrium prices, $\boldsymbol{p}$, for market $(\boldsymbol{u}, \boldsymbol{m})$ using the DPSV algorithm.

3. For each $i \in B$, compute γ_i w.r.t. prices $\boldsymbol{p}$, and set $m'_i \leftarrow 1 + \frac{c_i}{\gamma_i}$.

4. If $\boldsymbol{m}' = \boldsymbol{m}$ then output equilibrium allocations and **HALT**.
   Else, update $\boldsymbol{m}$ to $\boldsymbol{m}'$ and go to Step 2.

Let $\boldsymbol{p}^*$ and $\boldsymbol{m}^*$ be the equilibrium prices and moneys for the flexible budget market $\mathcal{M}$, and let $\boldsymbol{p}^{(k)}$ and $\boldsymbol{m}^{(k)}$ denote the prices and moneys computed by the algorithm in the k-th iteration, $k \geq 1$.

Lemma 35 $\boldsymbol{p}^{(k)}$ *and* $\boldsymbol{m}^{(k)}$ *are monotone increasing and are weakly dominated by* $\boldsymbol{p}^*$ *and* $\boldsymbol{m}^*$, *respectively.*

Proof :   We will use the following 2 facts. First, the DPSV algorithm maintains the following invariant throughout:

Invariant: W.r.t. current prices, $\boldsymbol{p}$, $(s, B \cup G \cup t)$ is a min-cut in network $N'(\boldsymbol{p})$.

Second, if $\boldsymbol{p}$ are equilibrium prices for money $\boldsymbol{m}$ and if $\boldsymbol{m}'$ is at least as large as $\boldsymbol{m}$ in each component, then the equilibrium prices for money $\boldsymbol{m}'$ cannot be smaller than $\boldsymbol{p}$ in any component.

Consider the following induction hypothesis:

- the algorithm given above maintains the Invariant throughout.

- $\boldsymbol{p}^{(k)}$ is monotone increasing (hence, for each agent i, γ_i is monotonically decreasing).

- $\boldsymbol{m}^{(k)}$ is monotone increasing.

It is easy to carry out this induction simultaneously for all 3 assertions.

Using the first assertion and Lemma 6, $\boldsymbol{p}^{(k)}$ is weakly dominated bounded by $\boldsymbol{p}^*$. Now, using the formula for money in flexible budget markets, it is easy to see that $\boldsymbol{m}^{(k)}$ is weakly dominated by $\boldsymbol{m}^*$.   □

Theorem 36 *Algorithm 3 converges to the equilibrium prices and moneys of market $\mathcal{M}$ in the limit.*

Proof :   We will use the following fact: for the linear case of Fisher's model, the analog of Lemma 4 holds, i.e., if $\boldsymbol{p}$ are equilibrium prices for money $\boldsymbol{m}$, then in network $N'(\boldsymbol{p})$, $(s, B \cup G \cup y)$ and $(s \cup B \cup G, t)$ must both be min-cuts (for a proof, see Lemma 5.2 in [Vaz07]).

Since $\boldsymbol{p}^{(k)}$ and $\boldsymbol{m}^{(k)}$ are monotone increasing and bounded, they must converge. Let $\boldsymbol{p}^{(0)}$ and $\boldsymbol{m}^{(0)}$ be their limit points. W.r.t. these prices and moneys, it must be the case that for each $i \in B$, $m_i = 1 + c_i/\gamma_i$ and $(s, B \cup G \cup t)$ and $(s \cup B \cup G, t)$ must both be min-cuts in the corresponding network (by the fact stated above). Using lemma 4 we get that $\boldsymbol{p}^{(0)}$ and $\boldsymbol{m}^{(0)}$ are equilibrium prices and moneys for market $\mathcal{M}$.   □

Finally, by Theorem 3 we get:

Corollary 37 *Algorithm 3 converges to the Nash bargaining solution for* **ADNB**.

B l_1-norm Does Not Suffice

Fundamental differences between complementary slackness conditions for an LP and the KKT conditions for a convex program (see Section 4 in [DPSV08] and Section 12) lead to new difficulties in designing an algorithm in the latter setting. The new algorithmic idea of balanced flows, introduced in [DPSV08], helps overcome these difficulties. Although this notion yields the desired efficient algorithm, the use of the l_2 norm in the potential function used for proving polynomial time termination makes the proofs quite difficult. [DPSV08] observe that the l_2

norm can be dispensed with for defining balanced flows and ask, "Can a polynomial running time be established for the algorithm using the alternative definition, thereby dispensing with the l_2 norm altogether? At present we see no way of doing this ... ."

In this section we answer this question in the negative by providing an infinite family of examples in which, under the natural l_1 norm-based potential function, the DPSV algorithm for Fisher's linear case, makes inverse exponentially small progress whereas the l_2 norm-based potential function makes inverse polynomial progress. We remark that the main idea underlying this family of examples yields similar examples for Stage I and Stage II of Algorithm 1.

We will define the example in terms of 2 parameters, δ and H, which will be set at the end. Assume $B = \{b_0, b_1, \ldots, b_{n-1}, b_n\}$ and $G = \{g_0, g_1, \ldots, g_{n-1}, g_n\}$. At the start of the phase, the only edges present in the network are (g_i, b_i), for $0 \leq i \leq n$. The money of the buyers are as follows:

$$m_0 = 1 + \delta, \quad \text{and for } 1 \leq i \leq n-1, \ m_i = \frac{\delta}{2^i}, \quad \text{and } m_n = H.$$

The prices of goods are as follows:

$$p_0 = 1, \quad \text{and for } 1 \leq i \leq n-1, \ p_i = \frac{\delta}{2^i}, \quad \text{and } p_n = H.$$

Hence, at the start of the phase, the surplus of b_0 is δ, and that of the rest of the buyers is 0.

We will set $\delta = 1$ and H to be a large number, say n. The phase starts with $I = \{b_0\}$ and $J = \{g_0\}$. Assume that at the end of iteration i, edge (g_i, b_{i-1}) enters the network, and as a result, b_i enters I and g_i enters J, for $1 \leq i \leq n$. The increment in price in each iteration is very small – this is easily arranged by choosing the right utilities u_{ij}'s.

To keep the description clean, let us assume the increments in price are all zero; the numbers can be easily modified by inverse exponential amounts to yield the desired outcome, even if the prices need to increase in each iteration. If so, at the end of all this, the surplus of b_i is

$$\frac{\delta}{2^{i+1}}, \quad \text{for } 0 \leq i \leq n-1,$$

and that of b_n is $\frac{\delta}{2^n}$.

Finally, in iteration $n + 1$, a very slight increase in x leads to set $\{g_n\}$ going tight. Observe that the reason for choosing H to be a large number is to ensure that this slight increase in x will not make a larger set go tight. Observe that $\Gamma(\{g_n\}) = \{b_n, b_{n-1}\}$. Now, the increase in the price of g_n needed for this is $\frac{\delta}{2^{n-1}}$. Since H is a large number and the increase in x is very small, the total increase in the prices of other goods is less than $\delta/2^{n-1}$.

In summary, the total increase in the l_1 norm of the surplus vector in this phase is an inverse exponential factor.