Spending Constraint Utilities, with Applications to the Adwords Market

Vijay V. Vazirani[*]

Abstract

The notion of a "market" has undergone a paradigm shift with the Internet – totally new and highly successful markets have been defined and launched by Internet companies, which already form an important part of today's economy and are projected to grow considerably in the future. Another major change is the availability of massive computational power for running these markets in a centralized or distributed manner.

In view of these new realities, the study of market equilibria, an important, though essentially non-algorithmic, theory within mathematical economics, needs to be revived and rejuvenated via an inherently algorithmic approach. Such a theory should not only address traditional market models but also define new models for some of the new markets.

We present a new, natural class of utility functions which allow buyers to explicitly provide information on their relative preferences as a function of the amount of money spent on each good. These utility functions offer considerable expressivity, especially in Google's Adwords market. In addition, they lend themselves to efficient computation, while still possessing some of the nice properties of traditional models.

Key words: Market equilibrium, Fisher's model, linear utilities, Adwords market, search engines, combinatorial algorithms, primal-dual algorithms, weak gross substitutability.

MSC 2000 Classification Codes: 91B24, 91B50.

OR/MS Classification Words: Analysis of Algorithms, Economics.

[*]Address: College of Computing, Georgia Institute of Technology, Atlanta, GA 30332–0280, E-mail: vazirani@cc.gatech.edu.

1 Introduction

General equilibrium theory, which produced such celebrated works as the Arrow-Debreu theorem and long enjoyed the status of crown jewel within mathematical economics, suffered from a serious shortcoming – other than a few isolated results, it was a non-algorithmic theory. Although a complexity-theoretic study of computing equilibria was initiated a while back by Megiddo [10] (see also the subsequent paper [11]), the real impetus for developing an algorithmic theory of market equilibria came only in this decade with the emergence of the area of algorithmic game theory. In turn, impetus for this area came from the emergence of the Internet as the quintessential new computational platform and the myriad of new issues of a strategic and computational nature raised by it.

With the emergence of new markets on the Internet, which already form an important part of today's economy and are projected to grow considerably in the future, and the availability of massive computational power for running these markets in a distributed or centralized manner, the need for developing an algorithmic theory of markets and market equilibria is quite apparent. Such a theory should not only address traditional market models but also define new models for some of the new markets. The latter task is not easy, since such a model should not only capture the idiosyncrasies of a new market in a simple manner but also have some of the nice properties of traditional models, such as existence and uniqueness of equilibria, and at the same time it should lend itself to efficient computation.

We attempt this task in the current paper. We define the notion of *spending constraint utility functions* within Fisher's market model [3]. We argue that the special case of decreasing step spending constraint utilities are well suited for expressing advertisers' desired allocations in the Adwords market, an innovative market which is run by search engine companies such as Google, Yahoo! and MSN. This multi-billion dollar market is the main source of revenues for Google and a major source of revenues for Yahoo!.

We give a polynomial time algorithm for computing an equilibrium for this case – our algorithm is made possible because this case satisfies the condition of *weak gross substitutability*, i.e., increasing the price of one good cannot result in a decreased demand of another good. This case shares several other desirable properties with the traditional model, some of which have been central to the area of general equilibrium theory. These include existence of equilibrium under certain mild conditions, uniqueness of equilibrium utilities and prices of goods, and the fact that equilibrium prices are rational with polynomial descriptions if all input parameters are rational.

The sequel to this paper, [7], continues the study of spending constraint utilities. For the case that spending constraint functions are continuous and strictly decreasing, [7] establish existence (using Brauwer's fixed point theorem) and uniqueness of equilibrium prices, and they show that this case also satisfies weak gross substitutability. They also use our algorithm as a subroutine to give an FPTAS for computing equilibrium prices for this case.

[7] also give a natural way of defining spending constraint utilities in the exchange model of Arrow and Debreu [2]. For the cases of step decreasing functions as well as continuous and strictly decreasing functions, they build on our algorithm polynomial time algorithm for Fisher's model to obtain FPTAS's. Furthermore, for continuous and strictly decreasing spending constraint functions, they show existence of equilibrium prices using the Kakutani fixed point theorem (Brauwer's theorem does not seem to yield the result for this model).

1.1 Comparison with concave and linear utilities

In Fisher's original model, buyers had additively separable, strictly concave utility functions. Such utility functions are considered especially useful in economics because they model the important condition of decreasing marginal utilities as a function of the amount of good obtained. Algorithmically though, such utility functions are not easy to deal with – in particular, they do not satisfy weak gross substitutability. A slight modification of these utilities, to additively separable, piecewise-linear, concave utilities have received much attention within algorithmic game theory and finding a good algorithm for them remains an outstanding open problem; see [9] for an early work giving an algorithm for the case of two traders in the exchange model.

Linear utility functions do satisfy weak gross substitutability and by exploiting this property, [6] gave the first polynomial time algorithm for computing an equilibrium for these utilities in Fisher's model. On the other hand, linear utility functions suffer from a number of serious shortcomings.

Spending constraint utility functions seem to offer a happy compromise between these two possibilities. They do satisfy weak gross substitutability and are amenable to efficient algorithms. On the other hand, they do not suffer from some of the more serious deficiencies of linear utility functions.

For simplicity of exposition, let us introduce spending constraint step utility functions as a way of rectifying the following two deficiencies of linear utility functions. First, under linear utility functions each buyer typically ends up spending her money on a single item; clearly, this is not the case with concave utility functions. To deal with this issue, let us generalize linear utility functions by specifying a limit on the amount of money buyer i can spend on good j.

Second, linear utility functions do not capture the important condition of buyers getting satiated with goods, e.g., as done by concave utility functions. To capture this, we generalize further – buyer i has several linear utility functions for good j, each with a specified spending limit. W.l.o.g. we may assume that these functions are sorted in decreasing order, and hence capture the condition that buyer i derives utility at decreasing rates on getting more and more of good j. As shown in Section 2, this set of functions can be more succinctly represented via a single decreasing-step function.

In Section 11 we make a further generalization – we assume that buyers have utility for money. Normally, in Fisher's model one does not assume this and as a consequence, at equilibrium all buyers are required to spend all their money. With the added assumption, the notion of equilibrium needs to be generalized appropriately. This generalization adds considerably to the expressivity of spending constraint step utility functions, as illustrated in the example given in Section 3.

1.2 The role of money in general equilibrium theory

The static nature of Walras' model leaves no place for the role of money. In real life though, money plays an vital role in the economy in two ways – it enables the exchange of goods and services over a period of time and it acts as a store of value for use in the future. Walras' model assumes that once agents determine the exchanges which they want to make, they happen instantaneously. In reality, there is a lag between selling and buying, with money bridging the gap in time.

Walras was well aware of this deficiency in his model but none of his attempts at rectifying it were satisfactory and over the next few decades, there was fervent debate on this issue. Perhaps the most successful attempt at integrating monetary and value theories was due to Patinkin [12]; see the insightful survey by Bridel [4]. Strictly speaking, our spending constraint utility function

fits better in the integrated theory than in Walras' theory.

1.3 An application to the Adwords market

When a user sends a query keyword to a search engine such as Google, he not only gets pages relevant to his query but also ads relevant to the keyword. These ads are sponsored by businesses (called advertisers below) who want to reach customers via Google. In the Adwords market run by Google, an advertiser selects keywords relevant to her business together with her bid for each keyword. Each bid represents the amount she is willing to pay to Google if her ad is shown along with search results to the corresponding keyword and moreover the user clicks on the ad. The advertiser also specifies her daily budget – the maximum amount that Google can charge her for each day – as well as spending limits on subsets of keywords.

It is not inconceivable that in the future, Google will simply be able to compute, in a centralized manner, prices for advertising on different keywords, instead of holding an elaborate auction. The question is how should advertisers provide information to Google so their ad gets displayed in the most effective manner and moreover, equilibrium prices of advertising on different keywords can also be efficiently computed by Google?

Let us list some criteria that a good utility should satisfy for this purpose:

1. The utility function should be expressive enough that the advertiser gets close to her "optimal allocation", i.e., one that is best for her long-term profit.

2. Equilibrium prices and allocations should be efficiently computable.

3. It should be easy for the advertiser to specify her utility function.

Observe that "optimal allocation" is not easy to define and it is not at all clear how it can be computed (perhaps by modeling the entire economy and then figuring out which allocation is best for this advertiser?). So, we will not attempt to formalize it and instead appeal to the reader's intuitive understanding of this notion.

Both linear and concave utility functions fail these criteria. With linear utility functions, on a typical day, a business will end up spending its entire advertising budget on only one of its desired keywords. Although concave utility functions are expressive enough to capture very complex requirements of an advertiser, equilibrium prices and allocations for these utility functions are not known to be computable in polynomial time.

Let us consider the third criterion above. Our tenet is that it would be very difficult for an advertiser to provide a function that captures the "utility" of a given set of keywords. Instead, it would be much easier for the business to partition the possible prices of keywords into a few ranges and for each range, specify how much it wants to spend on each keyword so as to have a profitable business. We illustrate our stance via an example in Section 3. We show how spending constraint step utility functions can provide businesses with a rich set of possibilities from which they can choose their desired allocations in Google's Adwords market.

1.4 Overview of algorithmic ideas

Spending constraint step utility functions generalize linear utility functions and our algorithm is obtained by generalizing the algorithm of [6]. Similar to [6], our algorithm is also based on the primal-dual paradigm, with allocations of goods playing the role of primal variables and prices

playing the role of dual variables. Our algorithm also starts with very low prices for the goods, so buyers have surplus money, and gradually raises prices until the surplus vanishes and the equilibrium is reached. This approach is made possible by the property of weak gross substitutability – on raising the price of one good, the demand of another good cannot go down, hence the need to decrease the price of the second good does not arise.

As stated in [6], there does not seem to be any natural linear programming formulation for computing equilibrium allocations for Fisher's linear case. However, there is a nonlinear convex program, given by Eisenberg and Gale [8], that does so. The algorithm of [6] uses the primal-dual paradigm not in its usual setting of LP-duality theory, but in the enhanced setting of convex programming and KKT conditions. The new difficulties raised by this setting and the manner in which they are circumvented are pointed out in [6].

Two main difficulties are the following. First, KKT conditions for nonlinear convex programs involve both primal and dual variables simultaneously in an equality constraint (obtained by assuming that one of the variables takes a non-zero value). On the other hand, equality constraints implied by complimentary slackness conditions for linear programs involve either primal or dual variables but not both. As a result, even though the dual growth process in [6] is greedy (prices of goods are non-decreasing), the primal objects (edges in the network N defined in Section 5.1) appear and disappear as the algorithm proceeds, thereby requiring a difficult accounting process for bounding the running time. Secondly, whereas other primal-dual algorithms satisfy complementary slackness conditions via a discrete process (one condition per iteration) the algorithm of [6] satisfies KKT conditions via a continuous process. This leads to a polynomial time, rather than a strongly polynomial, algorithm.

For our problem, we do not even know of a convex program that captures, as its optimal solution, equilibrium allocations – see Section 12 for a discussion of this issue. The combinatorial nature of the algorithm of [6] comes to our rescue. Indeed, such adaptability to variants and generalizations of the original problem – even when they do not admit linear or convex programming formulations – has been a major strength of combinatorial algorithms.

Our algorithm is obtained by simply using the essence of the primal-dual paradigm to the problem at hand. The first difficulty mentioned above is compounded further by the fact that at any prices, the optimal bundle of buyer i will involve *forced allocations*, i.e., at these prices, buyer i *necessarily* wants to spend a certain amount of her money on certain goods. However, as prices change, some of the forced allocations may become undesirable for buyer i and they need to be deallocated. The main new idea needed beyond [6] is in designing the algorithm in such a way that despite this backtracking it does not end up taking exponential time.

The potential function that measures progress of our algorithm is similar to that in [6]. Let m_i be the money spent by buyer i at some point in the run of the algorithm (w.r.t. a special allocation, as defined by a balanced flow; see Section 6). Thus, buyer i's surplus money is $\gamma_i = e_i - m_i$. Consider the following potential function:

$$\Phi = \gamma_1^2 + \gamma_2^2 + \ldots + \gamma_{n'}^2.$$

Our algorithm decreases this potential function by an inverse polynomial fraction in each phase, which can be implemented in strongly polynomial time. When Φ drops all the way to zero, equilibrium is reached. However, since Φ is a function of the initial money of the buyers, we only get a polynomial time algorithm. As in [6], the use of l_2 norm in our algorithm makes its proof difficult. Very recently, [13] has shown that l_1 norm-based potential functions do not suffice for establishing

polynomial running time of the algorithm of [6], and hence our algorithm.

Some of the notions introduced in [6] need to be generalized appropriately to our setting and as a result, their combinatorics becomes more involved. For some of the proofs though, the main idea is not very different from that in [6]. However, at the referees' suggestion, we have made this paper completely self-contained, rather than referring the reader to [6] for proof ideas.

2 Fisher's model and spending constraint utilities

Fisher's market model is the following. Let G be a set of divisible goods and B be a set of buyers, $|G| = n$, $|B| = n'$. Assume that the goods are numbered from 1 to n and the buyers are numbered from 1 to n'. Each buyer $i \in B$ comes to the market with a specified amount of money, say $e(i) \in \mathbf{Q}^+$ dollars, and we are specified the quantity, $q_j \in \mathbf{Q}^+$ of each good $j \in G$; throughout this paper we will assume w.l.o.g. that $q_j = 1$ for each $j \in G$, i.e., there is a unit amount of each good in the market. For each buyer i and good j we are specified a function $h_j^i : \mathbf{R}^+ \to \mathbf{R}^+$ which gives the utility that i derives as a function of the amount of good j that she receives. Her overall utility is additively separable over the goods. The problem is to find equilibrium prices, i.e., prices of goods such that if each buyer gets her optimal bundle, relative to these prices, for the money she has, the market clears exactly – there is no deficiency or surplus of any good.

This model has been studied under several different utility functions for buyers. Under the linear utility case, $h_j^i(x_{ij}) = u_{ij}x_{ij}$ where $u_{ij} \geq 0$ is a constant. Fisher had originally defined his model for the case that h_j^i is a strictly concave, differentiable function.

An easy way of describing *spending constraint step utility functions* is by contrast with piecewise-linear and concave functions. Suppose h_j^i is a piecewise-linear and concave function. Let r_j^i be the derivative of h_j^i; this will be a decreasing step function. Observe that function r_j^i specifies the rate at which i derives happiness on obtaining a unit amount of good j as a function of the amount of good j she has.

Under the spending constraint step function case, a decreasing step function f_j^i specifies the rate at which i derives happiness on obtaining a unit amount of good j *as a function of the amount of money she has spent on good* j. Next we note that once we know the price of a unit of good j, say p_j, we can obtain a function, g_j^i, that gives the utility derived by i as a function of the amount of money she spends on good j as follows:

$$g_j^i(x) = \int_0^x \frac{f_j^i(y)}{p_j} dy.$$

The contrast between the way utility is specified by h_j^i and r_j^i on the one hand and f_j^i and g_j^i on the other is worth understanding before proceeding further.

Next, let us formally define arbitrary spending constraint utility functions in Fisher's model. For $i \in B$ and $j \in G$, let $f_j^i : [0, e(i)] \to \mathbf{R}^+$ be the *rate function* of buyer i for good j; it specifies the rate at which i derives utility per unit of j received as a function of the amount of her budget spent on j. If the price of j is fixed at p_j per unit amount of j, then the function f_j^i/p_j gives the rate at which i derives utility per dollar spent, as a function of the amount of her budget spent on

j. Define $g_j^i : [0, e(i)] \rightarrow \mathbf{R}^+$ as follows:

$$g_j^i(x) = \int_0^x \frac{f_j^i(y)}{p_j} dy.$$

This function gives the utility derived by i on spending x dollars on good j at price p_j. This model satisfies the important property of weak gross substitutability, as will be shown formally in [7]. The reason is straightforward – raising the price of one good will only lead to an increased spending on any other good and hence an increased demand for the latter.

Each buyer also has utility for the part of her money that she does not spend. For $i \in B$, let $f_0^i : [0, e(i)] \rightarrow \mathbf{R}^+$ specify the rate at which i derives utility per dollar as a function of the amount she does not spend. If i returns with x dollars, the utility derived from this unspent money is given by

$$g_0^i(x) = \int_0^x f_0^i(y) dy.$$

By specifying suitable properties for f_j^i, the function g_j^i can be forced to have desirable properties. Thus, if f_j^i is continuous and monotonically decreasing, g_j^i will be strictly concave and differentiable. It is easy to see that for such functions, at any prices of the goods, there is a unique allocation that maximizes i's utility.

In this paper, we will deal with the case that the f_j^i's are decreasing step functions. If so, g_j^i will be a piecewise-linear and concave function. The linear version of Fisher's problem [3] is the special case in which each f_j^i is the constant function so that g_j^i is a linear function (in Fisher's original problem g_j^i's were concave functions), and each f_0^i is the zero function, so each buyer wishes to spend all her money. Given prices $\boldsymbol{p} = (p_1, \ldots, p_n)$ for all the goods, consider baskets of goods that make i happiest (there could be many such baskets). We will say that $\boldsymbol{p}$ are *market clearing prices* if after each i is given an optimal bundle, there is no deficiency or surplus of any good, i.e., the market clears. Observe that i's optimal bundle may contain unspent money.

We will call each step of f_j^i a *segment*. The set of segments defined in function f_j^i will be denoted $\text{seg}(f_j^i)$. Suppose one of these segments, s, has range $[a, b] \subseteq [0, e(i)]$, and $f_j^i(x) = c$, for $x \in [a, b]$. Then, we will define $\text{value}(s) = b - a$, $\text{rate}(s) = c$, and $\text{good}(s) = j$; we will assume that good 0 represents money. We will assume that for each segment s specified in the problem instance, $\text{rate}(s)$ and $\text{value}(s)$ are integral (this is w.l.o.g. since this can be ensured by appropriately scaling all numbers specified in the input). Let $\text{segments}(i)$ denote the set of all segments of buyer i, i.e.,

$$\text{segments}(i) = \bigcup_{j=0}^{n'} \text{seg}(f_j^i).$$

Let us assume that the given problem instance satisfies the following (mild) conditions:

- For each good, there is a potential buyer, i.e.,

$$\forall j \in A \; \exists i \in B \; \exists s \in \text{seg}(f_j^i) \; : \; \text{rate}(s) > 0.$$

- Each buyer has a desire to use all her money (to buy goods or to keep some unspent), i.e.,

$$\forall i \in B : \sum_{s \in \text{segments}(i),\ \text{rate}(s)>0} \text{value}(s) \geq e(i).$$

Theorem 1 *Under the conditions stated above, there exist unique market clearing prices.*

The proof of uniqueness is given in Section 4 and existence follows from the algorithm, which is the subject of the rest of the paper.

The following assumptions can be made w.l.o.g. (by suitable scaling):

- There is a unit amount of each good, i.e., $\forall j \in G, q_j = 1$.
- Each $e(i)$ and the value of each segment is integral.

Given nonzero prices $\boldsymbol{p} = (p_1, \ldots, p_n)$, we characterize optimal baskets for each buyer relative to $\boldsymbol{p}$. Define the *bang per buck* relative to prices $\boldsymbol{p}$ for segment $s \in \text{seg}(f_j^i), j \neq 0$, to be $\text{rate}(s)/p_j$. The bang per buck of segment $s \in \text{seg}(f_0^i)$ is simply $\text{rate}(s)$. Sort all segments $s \in \text{segments}(i)$ by decreasing bang per buck, and partition by equality into classes: $Q_1, Q_2, \ldots$. For a class Q_l, define $\text{value}(Q_l)$ to be the sum of the values of segments in it. At prices $\boldsymbol{p}$, goods corresponding to segments in Q_l make i equally happy, and those in Q_l make i strictly happier than those in Q_{l+1}.

Find k_i such that

$$\sum_{1 \leq l \leq k_i - 1} \text{value}(Q_l) < e(i) \leq \sum_{1 \leq l \leq k_i} \text{value}(Q_l).$$

By the conditions of Theorem 1, segments in Q_{k_i} have nonzero rate. At prices $\boldsymbol{p}$, i's optimal allocation must contain goods corresponding to all segments in $Q_1, \ldots, Q_{k_i-1}$, and a bundle of goods worth $e(i) - (\sum_{1 \leq l \leq k_i - 1} \text{value}(Q_l))$ corresponding to segments in Q_{k_i}. We will say that for buyer i, at prices $\boldsymbol{p}$, $Q_1, \ldots, Q_{k_i-1}$ are her *forced partitions*, Q_{k_i} is her *flexible partition*, and $Q_{k_i+1}, \ldots$ are her *undesirable partitions*.

3 The expressivity of spending constraint step utility functions

Typically buyers, whether they are individuals or businesses, have very complicated preferences. This is particularly true of businesses – their long term profit depends on numerous factors. Since capturing their exact utility function may not be feasible, one may have to settle for a good approximation. Two important criteria to be considered in choosing a utility function for a particular application are expressivity and computational complexity.

Let us consider the task outlined in Section 1.3, that of choosing a utility function for advertisers in Google's Adwords market. As argued in Section 1.3, neither linear nor concave utility functions are not suitable for this task. Via an elaborate example, we show below how rich the expressivity of spending constraint step utility functions is for this market.

Consider a business, B, that sells men's and women's clogs and assume for simplicity that it is only interested in the two keywords "men's clog" and "women's clog". Suppose its advertising budget on Google is \$100 per day. Using past information, B can compute its expected profit per click for each of these keywords; assume the expected profits are \$2 per click for "men's clog" and

$4 per click for "women's clog". Say that a keyword is *profitable* if its price per click is at most its expected profit per click. Thus the keyword "men's clog" ("women's clog") is profitable if its price per click is at most $2 ($4). If a keyword is profitable, then define its *rate of profit* to be the profit accrued per dollar spent on advertising. Thus if the price per click of "men's clog" ("women's clog") is p (q), then its rate of profit is $2/p$ ($4/q$).

Now assume that, depending on the actual prices per click of these two keywords, B's optimal allocation is the following:

- **If both keywords are profitable**
  - **and if the rate of profit of better keyword is at least twice that of the other,** then B wants to spend its entire budget on the better keyword.
  - **Otherwise,** it wants to spend $60 on the better keyword and $40 on the other keyword.
- **If neither keyword is profitable** then B wants to spend $20 on the more profitable keyword and nothing on the other keyword, just to have a presence in the market.
- **If only one keyword is profitable**
  - **and if the rate of profit on this keyword is at least 2** then B wants to spend its entire budget, i.e., $100, on this keyword.
  - **Otherwise,** it wants to spend $60 on this keyword and nothing on the unprofitable keyword.

It is easy to see that B can acquire this allocation using spending constraint step utilities defined via the following segments for the two keywords and for money.

- **"men's clog":** A segment of rate 2 and value $60 and a segment of rate 1 and value $40.
- **"women's clog":** A segment of rate 4 and value $60 and a segment of rate 2 and value $40.
- **money:** A segment of rate 1 and value $80 and a segment of rate 0 and value $20.

4 Uniqueness of equilibrium prices

In this section we prove uniqueness of equilibrium prices, as claimed in Theorem 1. Suppose there are two equilibrium prices $\boldsymbol{p}$ and $\boldsymbol{p}'$ with $\boldsymbol{p} \neq \boldsymbol{p}'$, i.e., $\exists j \; s.t. \; p_j \neq p'_j$. W.l.o.g. assume there is a good j such that $p'_j < p_j$. Let

$$\theta = \min_{j \in G} \frac{p'_j}{p_j}.$$

By assumption, $\theta < 1$. Let $S = \{j \in G \mid p'_j = \theta p_j\}$; this is the set of goods whose relative desirability increases the most if we switch from prices $\boldsymbol{p}$ to $\boldsymbol{p}'$.

Lemma 2 *Consider an arbitrary buyer i. Let D and D' be (any) optimal bundles for i relative to prices $\boldsymbol{p}$ and $\boldsymbol{p}'$. Let r and r' denote the amount of money spent by i on goods in S in these two bundles, respectively. Then, $r' \geq r$.*

Proof : Let Q_1 and Q_2 denote the set of segments, at prices $\boldsymbol{p}$, in i's forced and flexible allocations, respectively. Similarly, let Q_1' and Q_2' denote the set of segments, at prices $\boldsymbol{p}'$, in i's forced and flexible allocations, respectively.

Since goods in S become more desirable under prices $\boldsymbol{p}'$ as compared to prices $\boldsymbol{p}$, any segment $s \in Q_1$, whose good is in S, must also be in Q_1'. Now there are three cases w.r.t. segments in Q_2 and Q_2'. In each case, the reason given below shows that $r' \geq r$. We will use the following fact in the last two cases: if segment s corresponds to a good in S and segment s' to a good in $\overline{S}$ and if i prefers s to s' at prices $\boldsymbol{p}$ then she must prefer s to s' at prices $\boldsymbol{p}'$ as well.

1. No segment of Q_2 is in S. In this case, the result is obvious.

2. All segments of Q_2 are in S. Now, by the fact given above, either $Q_2 \subseteq Q_1'$ or $Q_2 = (Q_2' - Q_1)$ and $Q_1' \subseteq Q_1$; the latter case takes into consideration the fact that some segments may migrate from Q_1 to Q_2'. In either case, we are done.

3. In the remaining case, partition Q_2 into two sets, P_1 and P_2, depending on whether the corresponding good is or is not in S, respectively. Again, by the fact given above, either $P_1 \subseteq Q_1'$ or $P_1 = (Q_2' - Q_1)$.

The lemma follows. □

By Lemma 2, the buyers spend at least as much on goods in S at prices $\boldsymbol{p}'$ as they do at prices $\boldsymbol{p}$. Since both these prices are equilibrium prices, the total money spent on any good must equal its total value under the corresponding prices. Now, by definition of θ, goods in S have strictly less total value at prices $\boldsymbol{p}'$ than at prices $\boldsymbol{p}$, leading to a contradiction.

5 Basic terminology and Invariants for the algorithm

Our algorithm starts with very low prices on goods and iteratively raises them until equilibrium prices are reached. Three important considerations during the run of the algorithm are:

1. On termination, the algorithm must end with the correct forced allocations for all buyers w.r.t. to the equilibrium prices. The algorithm accomplishes this by making forced allocations and deallocations in a disciplined manner, as dictated by Invariant 1.

2. The algorithm must ensure that at intermediate points, the unique equilibrium price of any good is not exceeded. Invariant 2 helps ensure this.

3. Finally, the algorithm must ensure that rapid progress is made towards finding the equilibrium. The notion of balanced flows (Section 6) helps with this.

Let $\boldsymbol{p}$ denote the vector of current prices of all goods. At any intermediate point in the algorithm, certain segments are already allocated. By *allocating segment* s, $s \in \text{seg}(f_j^i), j \neq 0$, we mean allocating value(s) worth of good j to buyer i. The exact quantity of good j allocated will only be determined at termination, when prices are finalized. Assume that segment $s \in \text{seg}(f_j^i), j \neq 0$ has already been allocated to buyer i. By *deallocating segment* s we mean subtracting value(s) worth of good j from the total value of good j allocated to buyer i. In addition, at an intermediate point in the algorithm, some money may be returned to buyer i. Let returned(s), $s \in \text{seg}(f_0^i)$, denote

the amount of money returned to i, corresponding to segment s, where returned$(s) \leq$ value(s). If returned$(s) > 0$, then all segments $s' \in \text{seg}(f_0^i)$ having a higher rate must be fully returned, i.e., there is at most one partially returned segment for each buyer.

Let allocated(j) denote the total amount of money spent on good j, $j \neq 0$, i.e., the total value of j already allocated and let spent(i) denote the sum of the amount spent by buyer i on allocated segments and the amount of money already returned to her. Thus, when segment s is allocated, value(s) is added to allocated(j) and to spent(i), and when returned(s) money is returned to i, corresponding to segment $s \in \text{seg}(f_0^i)$, returned(s) is added to spent(i). Also, define the *money left over* with buyer i, $m(i) = e(i) - \text{spent}(i)$.

The set of allocated segments for each buyer i must satisfy:

Invariant 1: At current prices $\boldsymbol{p}$, let $Q_1, Q_2, \ldots$ be the sorted list of partitions of buyer i. There is an integer $t_i \geq 1$ such that all segments in partitions $Q_1, \ldots, Q_{t_i-1}$ are fully allocated and in addition, a (possibly empty) subset of segments in Q_{t_i} are also fully allocated, and no segments in partitions $Q_{t_i+1}, Q_{t_i+2}, \ldots$ are allocated. Furthermore, the total value of all fully allocated segments is $\leq e(i)$.

We will say that at prices $\boldsymbol{p}$, $Q_1, \ldots, Q_{t_i-1}$ are i's *allocated partitions* and Q_{t_i} is i's *current partition*. We will denote the latter by $Q^{(i)}$. The exact value of t_i depends on the order in which events happen in the algorithm; however, we will show that when the algorithm terminates, $t_i = k_i$.

Define the *current bang per buck* of buyer i, α_i, to be the bang per buck of partition $Q^{(i)}$. This is the rate at which i derives utility, per dollar spent, for allocations from $Q^{(i)}$ at current prices. Denote by $\boldsymbol{a}$, $\boldsymbol{s}$ and $\boldsymbol{m}$ the current allocations, amounts spent and left over money, i.e., (allocated(j), $j \in G$), (spent(i), $i \in B$) and $(m(i),\ i \in B)$, respectively. We will carry over all these definitions to sets, e.g. for a set $S \subseteq G$, $\boldsymbol{m}(S)$ will denote $\sum_{j \in S} m(j)$. For a set $S \subseteq G$, $\boldsymbol{p}(S)$ will denote the sum of prices of goods in S, i.e., $\boldsymbol{p}(S) = \sum_{j \in S} p_j$. Since we have assumed there is a unit amount of each good present, this is also the total value of all goods in set S.

5.1 The network N and tight sets

We next define network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$, which is a function of the current prices, allocations and amounts spent; see Figure 1. This network is defined over vertex set $G \cup B$ together with a source vertex, s, and a sink vertex, t. Corresponding to each buyer i and each segment $s \in Q^{(i)}$, the network contains the directed edge (j, i), where good$(s) = j$. The capacity of this edge, c_{ji}, equals value(s). It also contains directed edges (s, j), for each $j \in G$ with capacity $p_j -$ allocated(j), and directed edges (i, t), for each $i \in B$ with capacity $m(i)$. Throughout the algorithm, we will maintain the following:

Invariant 2: $(s, G \cup B \cup t)$ is a min-cut[1] in network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$.

Observe that as a consequence of Invariant 2, at current prices, it is possible to sell all goods to buyers who desire them in their optimal bundles. However, in general not all buyers can be given optimal bundles; some of them may have surplus money left over. The following lemma is obvious and provides the terminating condition for the algorithm.

Lemma 3 *Assume that Invariants 1 and 2 hold. Then, prices $\boldsymbol{p}$ are equilibrium prices iff $(s \cup G \cup B, t)$ is a min-cut in network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$.*

[1] This section assumes familiarity with the theory of cuts and flows, e.g., see [1, 5].

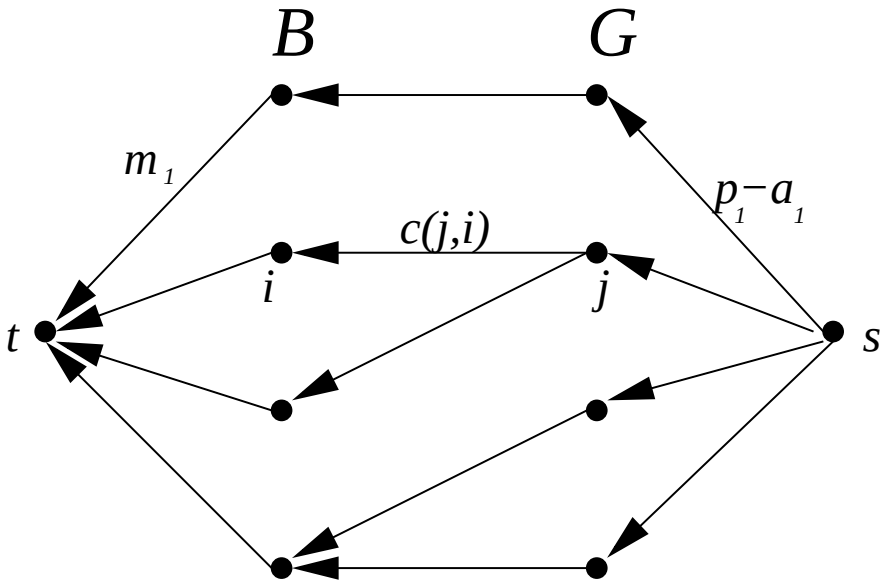

Figure 1: The network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$.

For $S \subseteq G$, define its *neighborhood in network* $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ to be

$$\Gamma(S) = \{i \in B \mid \exists j \in S \text{ with } (j, i) \in N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})\}.$$

For $G' \subseteq G$ and $B' \subseteq B$, define $c(G'; B')$ to be the sum of the capacities of all the edges from G' to B' in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. For $S \subseteq G$, define

$$\text{best}(S) = \min_{T \subseteq \Gamma(S)} \{\boldsymbol{m}(T) + c(S; \Gamma(S) - T)\},$$

and define bestT(S) to be a maximal subset of $\Gamma(S)$ that optimizes the above expression. Observe that best(S) is the capacity of the min-cut separating t from S in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. Also observe that if T_1 and T_2 optimize the above expression, then $i \in T_1 - T_2$ must satisfy $m(i) = c(S; i)$ (because otherwise $T_1 - \{i\}$ or $T_2 \cup \{i\}$ would be an even better set). Therefore, $T_1 \cup T_2$ also optimizes the above expression. Hence bestT(S) is unique. We can now give a characterization of Invariant 2 in terms of cuts in the network.

Lemma 4 *Network* $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ *satisfies Invariant 2 iff*

$$\forall S \subseteq G: \ \boldsymbol{p}(S) - \boldsymbol{a}(S) \leq \text{best}(S).$$

Proof : If network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ satisfies Invariant 2, it supports a max-flow of value $\boldsymbol{p}(G) - \boldsymbol{a}(G)$ that saturates all edges out of s. Since the flow going through set $S \subseteq G$ is $\boldsymbol{p}(S) - \boldsymbol{a}(S)$ and best(S) is the capacity of the min-cut separating t from S in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$, the inequality given above holds.

For the reverse direction, let $(s \cup G_1 \cup B_1, G_2 \cup B_2 \cup t)$ be a min-cut in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$, with $G_1, G_2 \subseteq G$ and $B_1, B_2 \subseteq B$; it has size $\boldsymbol{m}(B_1) + c(G_1, \Gamma(G_1) - B_1) + \boldsymbol{p}(G_2) - \boldsymbol{a}(G_2)$. Let $B_1' = B_1 \cap \Gamma(G_1)$. Then,

$$\text{best}(G_1) \leq \boldsymbol{m}(B_1') + c(G_1, \Gamma(G_1) - B_1) \leq \boldsymbol{m}(B_1) + c(G_1, \Gamma(G_1) - B_1).$$

Now, since $\boldsymbol{p}(G_1) - \boldsymbol{a}(G_1) \leq \text{best}(G_1)$, we get that

$$\boldsymbol{p}(G) - \boldsymbol{a}(G) \leq \boldsymbol{m}(B_1) + c(G_1, \Gamma(G_1) - B_1) + \boldsymbol{p}(G_2) - \boldsymbol{a}(G_2),$$

thereby proving that $(s, G \cup B \cup t)$ must also be a min-cut in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. Hence Invariant 2 holds. □

A nonempty set $S \subseteq G$ that satisfies the inequality in Lemma 4 with equality will be called a *tight set*. In addition, if there are buyers having zero left over money then we will say that the empty set is tight. We will define $\text{best}(\emptyset) = 0$ and $\text{bestT}(\emptyset)$ to be the set of all buyers with zero left over money. By the following lemma, if Invariant 2 holds, there is a unique maximal tight set.

Lemma 5 *Assume that Invariant 2 holds. If $S_1 \subseteq G$ and $S_2 \subseteq G$ are two tight sets, then $S_1 \cup S_2$ is also a tight set.*

Proof : Corresponding to a set $S \subseteq G$ modify network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ as follows: for each edge (s, j), $j \notin S$, make the capacity of this edge zero, leaving the rest of the edges unchanged. Call this the *S-network*. Since Invariant 2 holds, $(s, G \cup B \cup t)$ is a min-cut in this network. Recall that if S is a tight set, $\boldsymbol{p}(S) - \boldsymbol{a}(S) = \min_{T \subseteq \Gamma(S)} \{\boldsymbol{m}(T) + c(S; \Gamma(S) - T)\}$. Now, it is easy to see that S is a tight set iff under every max-flow in this network, there is no residual path from $j \in S$ to t.

Consider a max-flow in the $S_1 \cup S_2$-network. If there is a residual path from $j \in S_1 \cup S_2$ to t in this network, then either the S_1-network or the S_2-network would violate the residual path assertion stated above, contradicting tightness of the corresponding set. Hence, $S_1 \cup S_2$ is also a tight set. □

Corollary 6 *If Invariant 2 holds, the maximal tight set is unique.*

6 Balanced Flows

In this section, we will extend the notion of balanced flows from [6] to our more involved setting; the reason for the latter is that in our network N, edges between G and B have finite capacities. In retrospect, our generalization turns out to be a natural one.

As stated in Section 1.4, the potential function we will use for measuring the progress of our algorithm is the l_2^2-norm of the vector of surplus moneys of the buyers. This potential function follows naturally from the notion of a balanced flow. It enables us to record progress not only when the total surplus decreases but also when the surplus readjusts itself into a more favorable configuration that leads to a decrease in the total surplus in subsequent iterations.

Denote the current network, $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ by N, for short. We will assume that network N satisfies Invariant 2, i.e., $(\{s\}, G \cup B \cup \{t\})$ is a min-cut in N. Given a feasible flow f in N, let $R(f)$ denote the residual graph w.r.t. f. Define the *surplus* of buyer i, $\gamma_i(N, f)$, to be the residual capacity of the edge (i, t) with respect to flow f in network N, i.e., m_i minus the flow sent through the edge (i, t). The *surplus vector* is defined to be $\boldsymbol{\gamma}(N, f) := (\gamma_1(N, f), \gamma_2(N, f), \ldots, \gamma_{n'}(N, f))$. Let $\|v\|$ denote the l_2 norm of vector v. A *balanced flow* in network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ is a flow that minimizes $\|\boldsymbol{\gamma}(N, f)\|$. Clearly, a balanced flow must be a max-flow in N since augmenting a given flow can only lead to a smaller surplus vector.

Lemma 7 *All balanced flows in N have the same surplus vector.*

Proof : It is easy to see that if γ_1 and γ_2 are the surplus vectors w.r.t flows f_1 and f_2, then $(\gamma_1 + \gamma_2)/2$ is the surplus vector w.r.t the flow $(f_1 + f_2)/2$. Assume that $\boldsymbol{\gamma}_1$ and $\boldsymbol{\gamma}_2$ are distinct and both correspond to balanced flows, i.e., $\boldsymbol{\gamma}_1 \neq \boldsymbol{\gamma}_2$ and $\|\boldsymbol{\gamma}_1\| = \|\boldsymbol{\gamma}_2\|$. Since $\|.\|$ is strictly concave, $\|\boldsymbol{\gamma}_1 + \boldsymbol{\gamma}_2\|/2$ is even smaller, leading to a contradiction. □

We will denote the unique surplus vector of a balanced flow in network N by $\gamma(N)$. Clearly, the components of such a vector are "as equal as possible" among all surplus vectors corresponding to max-flows in N. The following property of balanced flows plays a critical role in the algorithm.

Property 1: If there is a path from node j to node i in $R(f) - \{s, t\}$, then $\boldsymbol{\gamma}_j(N, f) \geq \boldsymbol{\gamma}_i(N, f)$.

Theorem 8 *A maximum flow in N is balanced iff it satisfies Property 1.*

Proof : Let f be a balanced flow and let $\gamma_i(N, f) > \gamma_j(N, f)$ for some $i, j \in B$. Suppose, for the sake of contradiction, that there is a path from j to i in $R(f) - \{s, t\}$.

Since there is a path in $R(f) - \{s, t\}$ starting from vertex j, the capacity of (t, j) must be positive in $R(f)$. Also, since $\gamma_i(N, f) > 0$, the edge (i, t) has a positive capacity in $R(f)$. Now, the edges (t, j) and (i, t) concatenated with the path from j to i gives us a cycle with positive residual capacity in $R(f)$. Sending a circulation of positive value along this cycle will result in another max-flow in which the residual capacity of j is slightly larger and that of i is slightly smaller, i.e., the flow is more balanced. This contradicts the fact that f is a balanced flow.

To prove the other direction, we first show that a given max-flow f can be transformed to another max-flow f' by by a sequence of operations each of which changes only a pair of components of the surplus vector. Now $f' - f$ consists of circulations. Decompose this arbitrarily into cycles. Each of these cycles changes only a pair of components of the surplus vector.

Next, we observe that the l_2-norm of the surplus vector of a max-flow f satisfying Property 1 is locally optimum w.r.t. changes in pairs of components in the the surplus vector. This is so because by Property 1, any cycle in $R(f)$ can only send flow from a high surplus buyer to a low surplus buyer resulting in a less balanced flow. Now, since l_2-norm is a strictly concave function, any locally optimal solution is also globally optimal. Hence, a max-flow f satisfying Property 1 must be a balanced flow. □

6.1 Finding a balanced flow

We will show that the following algorithm, which uses a divide and conquer strategy, finds a balanced flow in the given network N on vertex set $\{s\} \cup G \cup B \cup \{t\}$ in polynomial time. As stated above, we will assume that this network satisfies Invariant 2, i.e., $(\{s\}, G \cup B \cup \{t\})$ is a min-cut in N. In addition, we may assume that $(\{s\} \cup G \cup B, \{t\})$ is not a min-cut, since the algorithm would have terminated otherwise.

Our algorithm does not simply partition N into two networks and finds balanced flows in each. It also needs to consider a certain flow that uses the "G-side" of one network and the "B-side" of the other and this makes the algorithm and proof more involved.

First, simultaneously and continuously reduce the capacities of all edges that go from B to t by equal additive amounts. As soon as the capacity of some edge becomes zero, don't decrease it any

more. Stop when the capacity of the cut $(\{s\} \cup G \cup B, \{t\})$ becomes the same as the capacity of the cut $(\{s\}, G \cup B \cup \{t\})$. Let the resulting network be N' and let f' be a max-flow in N'. Find a maximal $s-t$ min-cut in N', say (S,T), with $s \in S$ and $t \in T$; i.e., the min-cut that makes S maximal (standard cut theory shows that it is unique).

Case 1: $T = \{t\}$. Output f'; this will be a balanced flow in N.

Case 2: $T \neq \{t\}$. Let N_1 and N_2 be the subnetworks of N induced by $S \cup \{t\}$ and $T \cup \{s\}$, respectively. Let G_1 and B_1 be the subsets of G and B, respectively, induced by N_1. Similarly, let G_2 and B_2 be the subsets of G and B, respectively, induced by N_2. Let F be the set of edges that go from G_1 to B_2 – these edges are in the min-cut found. Send flow, say h, from s to t saturating all edges of F – clearly such a flow is unique. As a result of this flow, some of the capacity of edges from s to G_1 and B_2 to t will be used up. Subtract the used up capacities of these edges in N_1 and N_2 to obtain networks M_1 and M_2, respectively. Recursively find balanced flows, f_1 and f_2, in M_1 and M_2, respectively. Output the flow $f = h \cup f_1 \cup f_2$; this will be a balanced flow in N.

Lemma 9 *f is a max-flow in N.*

Proof : **Case 1:** $T = \{t\}$. Since the capacity of $(\{s\}, G \cup B \cup \{t\})$ is the same as the capacity of $(\{s\} \cup G \cup B, \{t\})$ in N', f' must saturate the former cut as well and hence must be a max-flow in network N as well.

Case 2: $T \neq \{t\}$. Because of the way M_1 and M_2 are defined, the union of the three flows, $f = h \cup f_1 \cup f_2$, will be a feasible flow in N. We show below that f is a max-flow as well. The structure of the argument is as follows. We start with a max-flow g in N. Using flow h and well-chosen subflows of g and f' in M_1 and M_2, respectively, we construct another max-flow, k, in N. We then argue that f, which is similar to k, must also be a max-flow in N.

Let g be any max-flow in N and let g_1 be the restriction of g to N_1. Note that $g_1 \cup h$ may not be a valid flow because some edges from s to G_1 may be over-saturated. If so, decrease flow g_1 appropriately to g_1' so $g_1' \cup h$ is a valid flow that saturates all edges from s to G_1. Let f_2' be the restriction of f' to network M_2; clearly, f_2' saturates all edes from s to G_2. Also, $h \cup f_2'$ is a valid flow in N.

Clearly, g_1' and f_2' are max-flows in M_1 and M_2, respectively, and $k = h \cup g_1' \cup f_2'$ is a max-flow in N. Hence, $f = h \cup f_1 \cup f_2$ is also a max-flow in N. □

Lemma 10 *f is a balanced flow in network N.*

Proof : We first show, by induction on the depth of recursion, that the max-flow output by the algorithm is a balanced flow in N. If the algorithm terminates in the first case, i.e., $T = \{t\}$, the surplus vector is precisely the amounts subtracted from capacities of edges running from B to t in going from N to N'. Clearly, this surplus vector makes components as equal as possible, thus minimizing its l_2 norm.

Next assume that the algorithm terminates in the second case. By Lemma 9, f is a max-flow; we will show that it satisfies Property 1 and is therefore a balanced flow. By the induction hypothesis, f_1 and f_2 are balanced flows in M_1 and M_2, respectively, and therefore Property 1 cannot be violated in either of these two networks.

Let R be the residual graph of N w.r.t. flow f; we need only show that paths in R that go from one part to the other do not violate Property 1. Since f saturates all edges of F, there are no

edges from G_1 to B_2 in R, and therefore there are no paths from $j \in B_1$ to $i \in B_2$. However, there may be paths going from $j \in B_2$ to $i \in B_1$ in R. Let $\gamma_i(f)$ denote the surplus of edge (i,t) w.r.t. flow f. We will show that for any two nodes $i \in B_1$ and $j \in B_2$, $\gamma_i(f) < \gamma_j(f)$, thereby establishing Property 1.

First observe that by the maximality (of the S-side) of the min-cut found in N', all nodes in B_2 have surplus capacity greater than 0 w.r.t. flow f' in N' (all nodes having surplus zero must be in B_1). Therefore, the same amount, α, say was subtracted from the capacity of each edge $(i,t), i \in B_2$, in going from network N to N'. We will show that $\gamma_i(f) > \alpha$ for each $i \in B_2$. A similar argument shows that $\gamma_i(f) \leq \alpha$ for each $i \in B_1$, thereby establishing our claim.

Let L be the set of vertices in B_2 having minimum surplus w.r.t. f. Let K be the set of vertices in G_2 that are reachable via an edge from L in R. Let F' be the set of edges from K to $B_2 - L$ in network N. If an edge of F' is not saturated in flow f_2 then there will be a residual path from $i \in L$ to $j \in B_2 - L$, thereby violating Property 1. Hence all edges of F' are saturated in f_2 and also in f.

Let $c(K)$ denote the sum of the capacities of all edges from s to vertices of K. Observe that all these edges are saturated in f'. Of this flow, at most $c(F')$ flow uses edges of F' and the rest, $c(K) - c(F')$ flow, must go via vertices of L. Let E_L denote the set of edges going from L to t. Let $c(L)$ and $c'(L)$ denote the sum of capacities of all edges in E_L in networks N and N', respectively. Since all nodes in B_2 have positive surplus w.r.t. flow f',

$$c'(L) > c(K) - c(F').$$

Since α is subtracted from all edges in E_L in going from network N to N',

$$c(L) = c'(L) + |L|\alpha.$$

The total surplus of the edges in E_L w.r.t. flow f is

$$c(L) - (c(K) - c(F')) = c'(L) + |L|\alpha - (c(K) - c(F')) > |L|\alpha.$$

Now, since all L edges in E_L have the same surplus, each has surplus greater than α. The lemma follows. □

Theorem 11 *The above-stated algorithm computes a balanced flow in network N using at most n max-flow computations.*

Proof : Clearly, the number of goods in the biggest piece drops by at least one in each iteration. Therefore, the depth of recursion is at most n. Next, observe that M_1 and M_2 are vertex disjoint, other than s and t, and therefore, the time needed to compute max-flows in them is bounded by the time needed to compute a max-flow in N. Hence, the total computational overhead is n max-flow computations. Finally, as shown in Lemma 10, the flow output by the algorithm is a balanced flow in N. □

Let $F' \subseteq F$ and let $N_{F'}$ be the network obtained from N by sending flow saturating all edges of F' and decreasing capacities of all edges accordingly. The following lemma will be used for justifying the manner in which forced allocations are made by the algorithm in the proof of Lemma 28.

Lemma 12 *The surplus vectors of balanced flows in N and $N_{F'}$ are the same, i.e., $\boldsymbol{\gamma}(N) = \boldsymbol{\gamma}(N_{F'})$.*

Proof : The lemma follows from the fact that in f all edges of F' are saturated, and, by Lemma 10, f is a balanced flow in N. □

7 The Main Algorithm

For ease of exposition and comprehension, we will first present the algorithm assuming that buyers have no utility for money. We will remove this restriction in Section 11.

The algorithm given below first determines a subgraph of the current partitions subgraph, C, called the *active subgraph*, and it increases the prices of goods only in the active subgraph. It does this in such a way that the active subgraph changes only when some important events (outlined in Step 4 in the algorithm) happen. Observe that if in the active subgraph buyer i has edges to goods j and j' due to segments $s \in \text{seg}(f^i_j)$ and $s' \in \text{seg}(f^i_{j'})$ then

$$\frac{\text{rate}(s)}{p_j} = \frac{\text{rate}(s')}{p_{j'}}, \quad i.e., \quad \frac{p_j}{p_{j'}} = \frac{\text{rate}(s)}{\text{rate}(s')}.$$

This suggests increasing the prices of goods in the active subgraph in such a way that the ratio of prices of any two goods remains unchanged. The algorithm accomplishes this by multiplying the current price, p_j, of each good j in the active subgraph by x, initializing $x = 1$, and raising x continuously.

A run of the algorithm is partitioned into *phases*, and each phase is partitioned into *iterations*, as defined below. In a phase, the prices of a subset, J, of the goods is increased – this set is well-chosen to ensure that the l_2 norm of the surplus vector decreases by at least an inverse polynomial factor in each phase. At the beginning of the phase, buyers having maximum surplus w.r.t. a balanced flow are identified as set I and an appropriate subset of the goods desired by these buyers is chosen to be J. After each iteration, I and J may grow. The phase ends when a subset of J goes tight.

Initialization: Execute the following steps to ensure that Invariants 1 and 2 start holding:

- Fix all prices at $1/n$. Since all goods together cost one dollar and all $e(i)$'s are integral, the initial prices are low enough that each buyer can afford all the goods. Each buyer's current partition will be her first partition. Recall that the value of each segment is assumed to be integral.
- In order to ensure that each good j has an interested buyer, i.e., has an incident edge in network N, compute α_i, the current bang per buck, for each buyer i at the prices fixed in the previous step. If good j has no incident edge, reduce its price to

$$p_j = \max_{i \in B} \max_{s \in \text{seg}(f^i_j)} \left\{ \frac{\text{rate}(s)}{\alpha_i} \right\}.$$

Phase: A phase starts with the computation of a balanced flow, say f, in the current network, $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. If the cut $(s \cup G \cup B, t)$ is saturated by f, then by Lemma 3 the current prices $\boldsymbol{p}$ are

equilibrium prices. If so, halt and output the current prices. Otherwise, let δ be the maximum surplus of buyers w.r.t. f. Initialize I to be the set of buyers having surplus δ. Go to **Step 1**.

- **Step 1:** Compute a balanced flow, say f, in the current network, $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. Let R be the corresponding residual graph. Determine the set of all buyers that have residual paths to buyers in the current set I (clearly, this set will contain all buyers in I). Update the new set I to be this set. Let P be the set of buyers having zero surplus w.r.t. flow f and let Q be the set of goods that can be reached from P using residual edges. (Clearly, Q is a tight set and raising the price of any good in this set will violate Invariant 2.) Let J' be the set of goods that have edges to I in N and let $J = J' - Q$. Let $I' \subseteq B - (P \cup I)$ be the set of buyers who have edges to $(J \cup Q)$ only.

- **Step 2:** Corresponding to each edge in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ which goes from Q to $(I \cup I')$, allocate goods; these are the *forced allocations*. (These edges would have appeared in the sets F in the recursive calls in the computation of the balanced flow.) Observe that because of Property 1, these edges must be saturated in f.

- **Step 3:** As a result of these forced allocations, there may be buyers $i \in (I \cup I')$ that have no more edges incident.

  If so, for each such buyer i, compute i's current partition and insert the corresponding edges in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. Go to **Step 1**.

- **Step 4:** At this point, call the network induced by $I \cup I'$ and J as the *active subgraph*. Multiply the current prices of all goods in J by variable x, initialize x to 1 and raise x continuously until one of the following 4 events happens. Observe that as soon as $x > 1$, buyers in $B - (I \cup I')$ are no longer interested in goods in J and all such edges can be dropped from the current partitions subgraph and N.

  - **Event 1:** For a buyer $i \in (I \cup I')$ and good $j \in G - J$, segment $s \in \text{seg}(f_j^i)$ enters i's current partition.

    If so, add directed edge (j, i) to network N and go to **Step 1**.
  - **Event 2:** A segment s enters buyer i's current partition, where $i \in B - (I \cup I')$, $s \in \text{seg}(f_j^i)$ for $j \in J$, and s is already allocated to i, i.e., the bang per buck of allocated segment s becomes equal to α_i.

    Deallocate segment s, i.e., subtract value(s) from allocated(j) and spent(i) and add directed edge (j, i) to network N. Go to **Step 1**.
  - **Event 3:** A subset $S \subseteq J$ goes tight.

    Terminate the current phase and start with the next phase.

Once the algorithm halts with equilibrium prices, equilibrium allocations are computed as follows. Compute a max-flow in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. A flow of f of good j to buyer i corresponds to f/p_j units of good j being given to buyer i. Similarly, the amount of good j corresponding to a forced allocation due to segment s is value(s)/p_j.

As stated above, the algorithm raises variable x continuously. This can be discretized as follows. Compute the minimum value of x at which each of the three events takes place; the minimum of

these is the event that happens first. For Events 1 and 2, the computation is straightforward. Let x^* be the value of x at which Event 3 happens. We give a procedure for computing x^* in Section 9.

The occurrence of Event 3 signifies the end of a *phase*. Within a phase, each execution of Step 1 will be called an *iteration*. Observe that an execution of Step 1 is triggered by Step 3 or Event 1 or 2.

Lemma 13 *Right after Step 1,* $(s \cup Q \cup P, (G - Q) \cup (B - P) \cup t)$ *is a min-cut in* $N(\boldsymbol{p}, \boldsymbol{s}, \boldsymbol{a})$.

Proof : Let $i \in P$ and $j \in (B - P)$. In the balanced flow, i has zero surplus and j has positive surplus. Therefore, by Property 1, there is no residual path from $i \in P$ to $j \in (B - P)$ in the residual graph of a balanced flow in $N(\boldsymbol{p}, \boldsymbol{s}, \boldsymbol{a})$. Therefore, all edges going from P to t and all edges going from Q to $B - P$ will saturated in this max-flow. Therefore, the capacity of these edges add up to the capacity of all edges going from s to Q. Now, since $(s, G \cup B \cup t)$ is a min-cut in $N(\boldsymbol{p}, \boldsymbol{s}, \boldsymbol{a})$ by Invariant 2, so is the cut $(s \cup Q \cup P, (G - Q) \cup (B - P) \cup t)$. □

Since the edges going from Q to $B - P$ include the edges going from Q to $I \cup I'$, on which forced allocations are made, we get:

Corollary 14 *Forced allocations corresponding to all edges connecting* Q *to* $(I \cup I')$ *can be made, i.e., there is sufficient value of goods available as well as sufficient amount of money available among buyers.*

Since Q is a tight set, increasing the price of any good in it will violate Invariant 2. Therefore, the prices of goods in $Q \cap J'$ cannot be increased. However, once forced allocations are made, the prices of goods in $J' - Q = J$ can be increased. Once the prices of goods in J are raised, these goods become inferior, compared to goods in Q, for buyers in $I \cup I'$,

We have assumed that for each segment s, value(s) is integral and therefore the capacities of edges in network N are integral. As a result the value of good allocated in each forced allocation is integral and hence $\boldsymbol{a}$ is an integral vector. Because of forced allocations, there may be a good $j \in Q$ that currently has no incident edge in network N. If so, by Invariant 2 it must be the case that $p_j = \text{allocated}(j)$.

Observe that under the spending constraint step utility functions, a segment $s \in \text{seg}(f_j^i)$ represents value(s) worth of good j; the exact amount of good j it represents will become clear only at the termination of the algorithm, once the equilibrium price of good j is determined.

Remark 15 *In contrast, under the usual piecewise-linear utility functions, each piece represents a certain quantity of a good and a forced allocation would have to allocate a specific amount of the good. However, as prices of the good change in the course of an iterative algorithm, the value of this allocation would keep changing and the surplus money of the buyer would keep changing as well. Our failure to find an algorithm to deal with these issues led us to define spending constraint utilities. In retrospect, since piecewise-linear utility functions do not satisfy weak gross substitutability, an iterative algorithm for dealing with piecewise-linear utility functions will quite possibly have to increase and decrease prices. Whether this can be accomplishied in polynomial time is unclear.*

8 Correctness of the algorithm

In this section we first show that the algorithm maintains Invariants 1 and 2 throughout and this helps us establish its correctness.

Lemma 16 *The algorithm maintains Invariant 1 throughout.*

Proof : The algorithm maintains Invariant 1 because it responds correctly to changes to Q_{t_i}, the current partition of buyer i, for each i. For each step and event in the main algorithm we state the changes made to Q_{t_i}.

- Step 2: If some segments in Q_{t_i} get allocated via forced allocations then these segments move into the new partition Q_{t_i-1}. The unallocated segments remain in Q_{t_i}.
- Step 3: If all segments in Q_{t_i} get allocated, then the next partition, Q_{t_i+1}, is made the current partition of i.
- Step 4: If the current partition of $i \in B - (I \cup I')$ has segments corresponding to goods in J, then as x is increased, these segments become inferior to the rest of her current segments and they move into a new partition Q_{t_i+1}.
- Event 1: If a segment moves from Q_{t_i+1} to Q_{t_i} for some buyer $i \in (I \cup I')$ then the corresponding edge is added to the current partitions subgraph.
- Event 2: If a segment s moves from Q_{t_i-1} to Q_{t_i} for buyer $i \in B - (I \cup I')$, then the algorithm deallocates this segment.

□

Lemma 17 *The algorithm maintains Invariant 2 throughout.*

Proof : The Initialization step clearly establishes Invariant 2 at the start of the algorithm. The algorithm does not raise the price of any good in tight set Q and stops raising prices of goods in J as soon as one of its subsets goes tight. Each of the remaining steps does not violate Invariant 2. Hence, it is maintained throughout. □

Theorem 18 *The algorithm terminates with equilibrium prices and allocations.*

Proof : By Lemmas 16 and 17, the algorithm maintains Invariants 1 and 2 throughout. Since it terminates when $(s \cup G \cup B, t)$ is a min-cut in network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$, by Lemma 3 the prices at this point must be equilibrium prices. Since it uses a max-flow in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ to compute allocations, they must be equilibrium allocations. □

9 Computing x^* via min-cuts in parametric networks

We will show how to compute x^*, the value of x at which Event 4 occurs, i.e., a new set goes tight in the active subgraph. The active subgraph is induced on the bipartition J, $(I \cup I')$. For simplicity, throughout this section we will denote $(I \cup I')$ by I.

Let $S^* \subseteq J$ denote the tight set. Throughout this section, $\boldsymbol{p}$ will denote prices at the beginning of the current phase, i.e., at $x = 1$. Network $W(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ is the subnetwork of $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ on $\{s\} \cup J \cup I \cup \{t\}$. In $W(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$, replace the capacities of edges (s, j), $j \in J$, by $(p_j \cdot x - \text{allocated}(j))$ to obtain the parametric network $W'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. By Invariant 2, at $x = 1$, $(s, J \cup I \cup t)$ is a min-cut in $W'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$.

Lemma 19 *The smallest value of x at which a new min-cut appears in $W'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ is given by*

$$x^* = \min_{\emptyset \neq S \subseteq J} \frac{\text{best}(S) + \boldsymbol{a}(S)}{\boldsymbol{p}(S)},$$

and the unique maximal set minimizing the above expression is S^*.

Proof : Let $x = \beta$ be the smallest value of x at which a new min-cut appears in $W'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. Let the min-cut maximizing the s side be $(s \cup J_1 \cup I_1, J_2 \cup I_2 \cup t)$. Since $W'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ satisfies Invariant 2 at $x = \beta$, for any set $S \subseteq J$,

$$\boldsymbol{p}(S) \cdot \beta - \boldsymbol{a}(S) \leq \text{best}(S), \quad i.e., \quad \beta \leq \frac{\text{best}(S) + \boldsymbol{a}(S)}{\boldsymbol{p}(S)}.$$

Since Invariant 2 holds and $(s \cup J_1 \cup I_1, J_2 \cup I_2 \cup t)$ is a min-cut in $W'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ at $x = \beta$, J_1 must be a tight set and therefore,

$$\boldsymbol{p}(J_1) \cdot \beta - \boldsymbol{a}(J_1) = \text{best}(J_1).$$

The lemma follows. □

Lemma 20 *The following hold:*

- *If $x \leq x^*$, then $(s, J \cup I \cup t)$ is a min-cut in $W'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$.*
- *If $x > x^*$, then for any min-cut $(s \cup J_1 \cup I_1, J_2 \cup I_2 \cup t)$ in $W'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$, $S^* \subseteq J_1$.*

Proof : By the definition of x^*, if $x \leq x^*$, $\forall S \subseteq J$: $\boldsymbol{p}(S) \cdot x - \boldsymbol{a}(S) \leq \text{best}(S)$. Therefore, by Lemma 4, Invariant 2 holds and hence $(s, J \cup I \cup t)$ is a min-cut in $W'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$.

Next, suppose that $x > x^*$, and consider a min-cut $(s \cup J_1 \cup I_1, J_2 \cup I_2 \cup t)$ in $W'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. First observe that $S^* \subseteq J_2$ contradicts the minimality of this cut: since $\boldsymbol{p}(S^*) \cdot x - \boldsymbol{a}(S^*) > \text{best}(S^*)$, a smaller cut results if S^* is moved into J_1, and $I_2 \cap \text{bestT}(S^*)$ is moved into I_1.

Let $S^* \cap J_1 = S_1$, $S^* \cap J_2 = S_2$, and suppose that $S_2 \neq \emptyset$. Observe that if $\Gamma(S_1) \cap \Gamma(S_2) = \emptyset$, then $\text{best}(S_1) + \text{best}(S_2) \leq \text{best}(S^*)$. To achieve a similar effect even if $\Gamma(S_1) \cap \Gamma(S_2) \neq \emptyset$ let us define for $S \subseteq I_2$:

$$\text{best}'(S) = \min_{T \subseteq \Gamma(S) - I_1} \{\boldsymbol{m}(T) + c(S; \Gamma(S) - I_1 - T) - c(S_1; T)\},$$

and let us define best$\mathrm{T}'(S)$ to be the (unique) maximal subset of $\Gamma(S)$ optimizing the above expression. Now observe that

$$\text{best}(S_1) + \text{best}'(S_2) \leq \text{best}(S^*).$$

Hence,

$$\text{best}(S_1) + \text{best}'(S_2) \leq x^* \cdot \boldsymbol{p}(S^*) - \boldsymbol{a}(S^*).$$

If $\text{best}'(S_2) < x \cdot \boldsymbol{p}(S_2) - \boldsymbol{a}(S_2)$, then a smaller cut can be found by moving S_2 into J_1, and moving best$\mathrm{T}'(S_2)$ from I_2 to I_1. Therefore,

$$\text{best}'(S_2) \geq x \cdot \boldsymbol{p}(S_2) - \boldsymbol{a}(S_2) > x^* \cdot \boldsymbol{p}(S_2) - \boldsymbol{a}(S_2).$$

Combining with the previous inequality, we get

$$\text{best}(S_1) < x^* \cdot \boldsymbol{p}(S_1) - \boldsymbol{a}(S_1),$$

which contradicts the definition of x^*. Therefore, $S_2 = \emptyset$ and hence $S^* \subseteq J_1$. □

For $i \in I$, denote the sum of capacities of edges incident at i in $W(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ by $c(i)$. Define $m'(i) = \min\{m(i), c(i)\}$, and $\boldsymbol{m'}$ to be the vector consisting of $m'(i), i \in I$. Observe that replacing $\boldsymbol{m}$ by $\boldsymbol{m'}$ in $W(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ does not change the min-cut or its capacity. Define $W''(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ to be the network obtained by replacing $\boldsymbol{m}$ by $\boldsymbol{m'}$ in $W'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. The reason for working with $\boldsymbol{m'}$ is that in $W''(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ the cut $(s \cup J \cup I_1, I_2 \cup t)$ has the same capacity as the cut $(s \cup J \cup I, t)$ for any partitioning of I into I_1 and I_2. This property will play a critical role in the next lemma.

Lemma 21 *Let $x = (\boldsymbol{m'}(I) + \boldsymbol{a}(J))/\boldsymbol{p}(J)$ and let the minimal min-cut in $W''(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ (i.e., the unique min-cut minimizing the s side) be $(s \cup J_1 \cup I_1, J_2 \cup I_2 \cup t)$. If $J_1 = I_1 = \emptyset$ then $x = x^*$ and $S^* = J$. Otherwise, $x > x^*$ and J_1 is a proper subset of J.*

Proof : Clearly, $x \geq x^*$. If the min-cut is $(s, J \cup I \cup t)$ then by Lemma 20, $x = x^*$. For the chosen value of x, $x \cdot \boldsymbol{p}(J) - \boldsymbol{a}(J) = \boldsymbol{m'}(I)$ and by the property of $\boldsymbol{m'}$ stated above, $\text{best}(J) = \boldsymbol{m'}(I)$. Therefore $\text{best}(J) = x^* \cdot \boldsymbol{p}(J) - \boldsymbol{a}(J)$, and hence $S^* = J$.

If $(s, J \cup I \cup t)$ is not a min-cut, then by Lemma 20, $x > x^*$. Suppose $J_1 = J$ and the min-cut is $(s \cup J \cup I_1, I_2 \cup t)$. By the property stated above, the capacity of this cut is $\boldsymbol{m'}(I)$. For the chosen value of x, the capacity of $(s, J \cup I \cup t)$ is also the same, thereby contradicting the fact that it is not a min-cut. Hence J_1 is a proper subset of J. □

Theorem 22 *x^* and S^* can be found using at most n max-flow computations.*

Proof : Let $x = (\boldsymbol{m'}(I) + \boldsymbol{a}(J))/\boldsymbol{p}(J)$ and compute a min-cut in $W''(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. If $(s, J \cup I \cup t)$ is a min-cut in $N''(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$, then by Lemma 21 $x^* = x$ and $S^* = J$. Otherwise, $x > x^*$. Let $(s \cup J_1 \cup I_1, J_2 \cup I_2 \cup t)$ be a min-cut in $W''(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. By Lemmas 20 and 21, $S^* \subseteq J_1 \subset J$. Therefore, it is sufficient to recurse on the network restricted to $(J_1, \Gamma(J_1))$ (of course $\boldsymbol{m'}$ will have to be recomputed for this restricted network). □

10 Running time of the algorithm

Observe that the algorithm keeps raising prices of goods monotonically, and this provides us with a natural measure of progress – the difference between total money possessed by buyers and the sum of the prices of all goods. In this section, we will establish polynomial running time for our algorithm.

Let M denote the total amount of money possessed by the buyers at the start of the algorithm and U denote the largest rate of a segment. Let $\Delta = nU^n$. Let z denote the total number of segments in all utility functions specified in the input.

Lemma 23 *At the termination of a phase, the prices of goods in the tight set are rational numbers with denominators* $\leq \Delta$.

Proof : Let $S \subseteq G$ be the newly tight set at the termination of a phase and let $T = \text{bestT}(S)$. If $T = \emptyset$, then all goods of S are fully allocated and for each such good $j \in S$, $p_j = \text{allocated}(j)$, which is integral. Otherwise, by definition of tight set,

$$\boldsymbol{p}(S) - \boldsymbol{a}(S) = \boldsymbol{m}(T) - c(S; \Gamma(S) - T).$$

Consider the current partitions subgraph induced on the bipartition (S, T). If this is not one connected component, let (S', T') be a connected component. Then, the equation given above must hold after replacing S and T by S' and T', because otherwise some connected component of (S, T) will fail to satisfy Invariant 2.

Therefore, we may assume w.l.o.g. that (S, T) is connected (otherwise we prove the lemma for each connected component of this graph). Let $j \in S$. Pick a subgraph in which j can reach all other vertices $j' \in S$. Clearly, at most $2|S| \leq 2n$ edges suffice. If j reaches j' with a path of length $2l$, then $p_{j'} = ap_j/b$ where a and b are products of the l rates of the corresponding segments. Since alternate edges of this path contribute to a and b, we can partition the rates in this subgraph into two sets such that a and b use rates from distinct sets. Now it is easy to show that $\boldsymbol{p}(S) = p_j c/d$ where $c \leq \Delta$. On the other hand, $\boldsymbol{p}(S) = \boldsymbol{m}(T) + \boldsymbol{a}(S) - c[S; \Gamma(S) - T]$. Since each term on the right hand side is integral, so is $\boldsymbol{p}(S)$. Therefore,

$$p_j = \boldsymbol{p}(S)d/c,$$

hence proving the lemma. □

Corollary 24 *Consider two phases P and P', not necessarily consecutive, 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* $\geq 1/\Delta^2$.

Proof : Let the prices of j at the end of P and P' be p/q and r/s, respectively. Clearly, $r/s > p/q$. By Lemma 23, $q \leq \Delta$ and $r \leq \Delta$. Therefore the increase in price of j,

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

□

Lemma 25 *The total number of iterations in a phase is bounded by z.*

Proof : We will show that in a phase, corresponding to each segment s, there can be at most one invocation of Step 1. Suppose Step 1 is invoked because the edge corresponding to segment s is added to network N (this could happen in Step 3 or in Event 1). Assume $s \in \text{seg}(f_j^i)$, for some good j. Clearly at this point, $i \in I$ and i will not move to $B - I$ in the current phase. Therefore, segment s will not be deallocated in the current phase.

Let buyer $i \in B - (I \cup I')$ and segment $s \in \text{seg}(f_j^i)$. Assume that s is deallocated; clearly, $j \in J$ at this stage. If i does not move into $I \cup I'$ in this phase, then s is not reallocated in this phase.

Next, assume that buyer i moves into $I \cup I'$ in the current phase. If this happens before any further price increase, then i must move into I, s is in i's current partition and there will not be any more invocations of Step 1 due to s in the current phase. On the other hand, if i moves into $I \cup I'$ after a price increase, then s is no longer in i's current partition and this segment will never be considered again in the current phase. □

Lemma 26 *If f and f^* are respectively a feasible and a balanced flow in $N(\boldsymbol{p})$ such that $\gamma_i(\boldsymbol{p}, f^*) = \gamma_i(\boldsymbol{p}, f) - \delta$, for some $i \in B$ and $\delta > 0$, then $\|\boldsymbol{\gamma}(\boldsymbol{p}, f)^*\|^2 \leq \|\boldsymbol{\gamma}(\boldsymbol{p}, f)\|^2 - \delta^2$.*

Proof : Suppose we start with f and get a new flow f' by decreasing the surplus of i by δ, and increasing the surpluses of some other buyers in the process. We show that this already decreases the l_2 norm of the surplus vector by δ^2 and so the lemma follows.

Consider the flow $f^* - f$. Decompose this flow into flow paths and circulations. Among these, augment f with only those that go through the edge (i, t), to get f'. These are either paths that go from s to i to t, or circulations that go from i to t to some i_l and back to i. Then $\gamma_i(f') = \gamma_i(f^*) = \gamma_i(f) - \delta$ and for a set of vertices $i_1, i_2, \cdots, i_k$, we have $\gamma_{i_l}(f') = \gamma_{i_l}(f) + \delta_l$, s.t. $\delta_1, \delta_2, \cdots, \delta_k > 0$ and $\sum_{l=1}^{k} \delta_l \leq \delta$. Moreover, for all l, there is a path from i to i_l in $R(\boldsymbol{p}, f^*)$. Since f^* is balanced, and satisfies Property 1, $\gamma_i(f') = \gamma_i(f^*) \geq \gamma_{i_l}(f^*) \geq \gamma_{i_l}(f')$.

By Lemma 27, $\|\boldsymbol{\gamma}(\boldsymbol{p}, f')\|^2 \leq \|\boldsymbol{\gamma}(\boldsymbol{p}, f)\|^2 - \delta^2$ and since f^* is a balanced flow in $N(\boldsymbol{p})$, $\|\boldsymbol{\gamma}(\boldsymbol{p}, f^*)\|^2 \leq \|\boldsymbol{\gamma}(\boldsymbol{p}, f')\|^2$. □

Lemma 27 *If $a \geq b_i \geq 0, i = 1, 2, \ldots, n$ and $\delta \geq \sum_{j=1}^{n} \delta_j$ where $\delta, \delta_j \geq 0, j = 1, 2, \ldots, n$, then $\|(a, b_1, b_2, \ldots, b_n)\|^2 \leq \|(a + \delta, b_1 - \delta_1, b_2 - \delta_2, \ldots, b_n - \delta_n)\|^2 - \delta^2$.*

Proof :

$$(a + \delta)^2 + \sum_{i=1}^{n} (b_i - \delta_i)^2 - a^2 - \sum_{i=1}^{n} {b_i}^2 \geq \delta^2 + 2a(\delta - \sum_{i=1}^{n} \delta_i) \geq \delta^2.$$

□

Let N_0 denote the network at the beginning of a phase. Assume that the phase consists of k iterations, and that N_t denotes the network at the end of iteration t. Let f_t be a balanced flow in N_t and I_t be the set of buyers in the active subgraph in N_t, for $0 \leq t \leq k$.

Lemma 28 *f_t is a feasible flow in N_{t+1}, for $0 \leq t < k$.*

Proof : First observe that by Lemma 12, the forced allocations do not change the surplus vector. Furthermore, each of the following actions can only lead to a network that supports an augmented max-flow:

- Raising the prices of goods in J.
- Adding an edge, as required in Step 3 or Event 1.
- Deallocating a segment, as required in Event 2.

The lemma follows. □

Lemmas 26 and 28 yield:

Corollary 29 *$\|\boldsymbol{\gamma}(N_t)\|$ is monotonically decreasing with t.*

Let δ_t denote the minimum surplus of a buyer in I_t in network N_t, for $0 \leq t \leq k$; clearly, $\delta_0 = \delta$ and $\delta_k = 0$.

Lemma 30 *If $\delta_{t-1} > \delta_t$ then there exists an $i \in I_{t-1}$ such that $\gamma_i(\boldsymbol{p}_{t-1}) - \gamma_i(\boldsymbol{p}_t) \geq \delta_{t-1} - \delta_t$.*

Proof : Consider the residual network $R(\boldsymbol{p}_t, f)$ corresponding to the balanced flow computed at the end of iteration t. By the definition of I_t, every vertex $v \in I_t \setminus I_{t-1}$ can reach a vertex $i \in I_{t-1}$ in $R(\boldsymbol{p}_t, f)$ and therefore, by Property 1, $\gamma_v(\boldsymbol{p}_t) \geq \gamma_i(\boldsymbol{p}_t)$. This means that the minimum surplus δ_t is achieved by a vertex i in I_{t-1}. Hence the surplus of vertex i is decreased by at least $\delta_{t-1} - \delta_t$ during iteration t. □

Lemma 31 *If $\delta_t > \delta_{t+1}$ then $\|\boldsymbol{\gamma}(N_t)\|^2 - \|\boldsymbol{\gamma}(N_{t+1})\|^2 \geq (\delta_t - \delta_{t+1})^2$, for $0 \leq t < k$.*

Proof : By Lemma 30, if $\delta_t > \delta_{t+1}$ then there is a buyer i whose surplus drops by $\delta_t - \delta_{t+1}$ in going from f_t to f_{t+1}. By Lemma 28, f_t is a feasible flow in N_{t+1}. Finally, by Lemma 26 we get the desired conclusion. □

Lemma 32 $$\|\boldsymbol{\gamma}(N_0)\|^2 - \|\boldsymbol{\gamma}(N_k)\|^2 \geq \frac{\delta^2}{z}.$$

Proof : The left hand side can be written as a telescoping sum in which each term is of the form $\|\boldsymbol{\gamma}(N_t)\|^2 - \|\boldsymbol{\gamma}(N_{t+1})\|^2$. By Corollary 29, each of these terms is nonnegative.

Consider only those terms in which the difference $\delta_t - \delta_{t+1} > 0$. Their sum is minimized when all these differences are equal. Now using Lemma 31 and the fact that $\delta_0 = \delta$ and $\delta_k = 0$, we get that

$$\|\boldsymbol{\gamma}(N_0)\|^2 - \|\boldsymbol{\gamma}(N_k)\|^2 \geq \frac{\delta^2}{k}.$$

By Lemma 25, $k \leq z$. The lemma follows. □

Lemma 33 *In a phase, the l_2^2-norm of the surplus vector drops by a factor of*

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

Proof : Lemma 32 and the fact that $\|\boldsymbol{\gamma}(N_0)\|^2 \leq n\delta^2$,

$$\|\boldsymbol{\gamma}(N_k)\|^2 \leq \|\boldsymbol{\gamma}(N_0)\|^2 - \frac{n\delta^2}{zn} \leq \|\boldsymbol{\gamma}(N_0)\|^2 - \frac{\|\boldsymbol{\gamma}(N_0)\|}{zn}$$

$$\leq \|\boldsymbol{\gamma}(N_0)\|^2 \left(1 - \frac{1}{zn}\right).$$

The lemma follows. □

Theorem 34 *The algorithm finds equilibrium prices and allocations for spending constraint step utility functions in Fisher's model using*

$$O\left(n^2 z^2 (\log n + n \log U + \log M)\right)$$

max-flow computations (see the second paragraph of this section for definitions of these parameters).

Proof : By Lemma 33, the square of the surplus vector drops by a factor of half after $O(zn)$ phases. At the start of the algorithm, the square of the surplus vector is at most M^2. Once its value drops below $1/\Delta^4$, by Corollary 24, equilibrium prices have been attained. Therefore the number of phases is bounded by

$$O(nz \log(\Delta^4 M^2) \;=\; O(nz \log n + n \log U + \log M).$$

By Lemma 25 each phase consists of z iterations and by Theorem 22 each iteration requires n max-flow computations. The theorem follows. □

11 Buyers have utility for money

Finally, we will assume that buyers have utility for money, given by step utility function f_0^i for each buyer $i \in B$. We note that segments corresponding to these utility functions will not appear as edges in the current partitions subgraph or the network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$.

As prices of goods are raised, the current bang per buck, α_i, of each buyer $i \in (I \cup I')$ keeps decreasing. If for some buyer $i \in (I \cup I')$, α_i decreases to the point where she is equally happy leaving with money corresponding to segment $s \in seg(f_0^i)$, then the algorithm will need to return money corresponding to s before it can raise the prices of goods any more.

As described below, the current phase may terminate even before money corresponding to s can be fully returned to i. If so, we will say that buyer i has a *partially returned segment* s; in this case, $0 < \text{returned}(s) < \text{value}(s)$. When i returns to the active subgraph in a subsequent phase, the algorithm first attempts to return the rest of value(s) to i.

The following event is executed as the first event in Step 4 of the main algorithm.

- **Event 0:** There is a buyer $i \in (I \cup I')$ with rate$(s) = \alpha_i$ for $s \in seg(f_0^i)$, and moreover, returned$(s) <$ value(s).

  Increase returned(s) continuously until one of two events happens:

  - **Event 0(a):** A set $S \subseteq J$ goes tight. (Observe that for a set $S \subseteq J$ such that $i \in$ bestT(S), best(S) is decreasing as returned(s) is raised; since $m(i)$ is decreasing.)

    If so, terminate the current phase and start with the next phase.
  - **Event 0(b):** returned$(s) =$ value(s), i.e., the money corresponding to segment s has been fully returned.

    Go to Step 4 and continue raising the prices of goods in J.

Let y^* denote the value of returned(s) at which Event 0(a) occurs. Next, we give a procedure for determining whether Event 0(a) or Event 0(b) occurs, and in the former case, we will give a method for computing y^*. Compute $\boldsymbol{p}'$, the prices of all goods at the moment Event 0 occurs. Let $\boldsymbol{a}$ denote all forced allocations made so far. Compute the money returned to buyers; for i assume, for the purpose of this procedure, that segment s is fully returned. Let $\boldsymbol{s}'$ denote the vector of money spent. As in Section 9, we will let I denote $I \cup I'$.

Construct network $W(\boldsymbol{p}', \boldsymbol{a}, \boldsymbol{s}')$ on vertices $\{s\} \cup J \cup I \cup \{t\}$ and find a maximal min-cut in it. If $(s, J \cup I \cup t)$ is the maximal min-cut, then Event 0(b) occurs, i.e., the entire money corresponding to segment s can be returned to i without a set going tight. Next assume that the maximal min-cut in the network is $(s \cup J_1 \cup I_1, J_2 \cup I_2 \cup t)$, with $J_1 \neq \emptyset$. If so, Event 0(a) occurs. Clearly, the procedure stated above uses only one max-flow computation.

Lemma 35 *If Event 0(a) occurs,*

$$y^* = \text{value}(s) - (\boldsymbol{p}'(J_1) - \boldsymbol{a}(J_1) - (\boldsymbol{m}(I_1) + c(J_1; \Gamma(J_1) - I_1)).$$

Proof : Let Situation 1 be the situation in which the entire money corresponding to segment s returned to i. Let Situation 2 be the scenario in which the money returned to i corresponding to segment s is precisely y^*; let C be the maximal min-cut in Situation 2. Observe that i will be on the s-side of cut C since it has spent all its money. Therefore the capacity of C in Situation 1 will be smaller by exactly value$(s) - y^*$. In addition, the difference in capacity of any other cut, C' in the two situations is at most value$(s) - y^*$.

Let cap_1 and cap_2 denote functions giving the capacities of cuts in Situation 1 and 2, respectively. Let $\beta =$ value$(s) - y^*$. Then, $\text{cap}_2(C') \geq \text{cap}_2(C)$, $\text{cap}_1(C) = \text{cap}_2(C) - \beta$, and $\text{cap}_1(C') \geq \text{cap}_2(C') - \beta$. Therefore, $\text{cap}_1(C') \geq \text{cap}_1(C)$. Therefore, C is a maximal min-cut in Situation 1 as well.

Since Invariant 2 holds, $(s, J \cup I \cup t)$ will also be a min-cut in Situation 2. Therefore the difference in the capacities of $(s \cup J_1 \cup I_1, J_2 \cup I_2 \cup t)$ in the two situations is precisely $\boldsymbol{p}'(J_1) - \boldsymbol{a}(J_1) - (\boldsymbol{m}(I_1) + c(J_1; \Gamma(J_1) - I_1))$. This yields the expression for y^* given in the statement of this Lemma. □

We prove below a lemma that is analogous to Lemma 23 for the enhanced model.

Lemma 36 *If Event 0(a) occurs, the prices of goods in the tight set are rational numbers with denominators* $\leq \Delta$.

Proof : Let $S \subseteq G$ be the newly tight set at the termination of the phase when Event 0(a) occurs and let $T = \text{bestT}(S)$. Let s be the segment that was being returned to buyer i when this happened. Clearly, $i \in T$ and the current partitions subgraph induced on (S, T) must be a single connected component (otherwise the component not containing i would contain a tight set even before s was returned). Pick a spanning tree, τ, in (S, T).

Observe that when rate(s) became equal to α_i, for any edge (i, j) incident at i,

$$p_j = \frac{\text{rate}(i,j)}{\text{rate}(s)},$$

where, by a slight abuse of notation, we are using rate(i, j) to denote the rate of the segment represented by the edge connecting i to j. Similarly, if j' is reached via the path i, j, i', j' in τ, then

$$p_{j'} = \frac{\text{rate}(i,j) \cdot \text{rate}(i',j')}{\text{rate}(s) \cdot \text{rate}(i',j)}.$$

Therefore, the denominator of $p_j,\ j \in T$ is the product of rates of at most n segments and hence is bounded by U^n, which in turn is bounded by Δ. □

If Event 0(b) occurs while returning money corresponding to segment s, then this segment will never be considered again, since the bang per buck of s remains unchanged but α_i can only decrease as the algorithm proceeds. Hence the number of occurrences of Event 0(b) is bounded by the number of segments in functions f_0^i, for all i, which in turn is bounded by z. Now, it is easy to see that the running time bound established in Theorem 34 holds for the enhanced model as well, as long as z is taken to be the total number of segments in all utility functions specified in the input, including those for money.

12 Discussion

The remarkable convex program given by Eisenberg and Gale [8] captures, as its optimal solution, equilibrium allocations for Fisher's linear model. Some of the basic properties of Fisher's linear case can be established in a simple manner via this program. These include the existence of an equilibrium under certain mild conditions, the uniqueness of equilibrium utilities and prices of goods, and the fact that equilibrium prices are rational (if all input parameters are rational) and have polynomially bounded descriptions.

In this paper, we have established the above-stated properties for spending constraint step utility functions in Fisher's model; uniqueness is shown in Section 4 and the other properties follow from our algorithm. It is natural, therefore, to ask if there is a convex program that captures equilibrium allocations for these utility functions.

We believe that the answer to this question should be "yes". In our experience, non-trivial polynomial time algorithms for problems are rare and happen for a good reason – a deep mathematical structure intimately connected to the problem. Observe that a convex program with the same structure as the Eisenberg-Gale program is not the right answer to this problem, since in our model utilities of buyers are not only a function of allocations but also of prices of goods, whereas in

the Eisenberg-Gale program, prices are Lagrangian variables corresponding to packing constraints occurring in the program.

An important open question regarding Fisher's linear case, which applies to spending constraint step utilities as well, is whether there is a strongly polynomial algorithm for computing the equilibrium. In particular, if such an algorithm is found for the former question, it will be interesting to determine if it generalizes naturally to our setting as well.

An important step toward handling concave utility functions may be obtaining a polynomial time algorithm for the case of piecewise-linear, concave utilities. Such utility functions need not satisfy weak gross substitutability. Our algorithm for spending constraint step utility functions may help finesse this difficulty via the following scheme given in [7]. Let f_{ij} be the piecewise-linear, concave utility function of buyer i for good j; f_{ij} is a function of x_{ij}, the allocation of good j to buyer i. Let g_{ij} be the derivative of f_{ij}; clearly, g_{ij} is a decreasing step function. Suppose the price of good j (not necessarily equilibrium price) is fixed at p_j. Define

$$h_{ij}(y_{ij}) = g(\frac{y_{ij}}{p_j}),$$

where y_{ij} denotes the amount of money spent by i on good j. Observe that function h_{ij} gives the rate at which i derives utility per unit of j received as a function of the amount of money spent on j. Hence h_{ij} is precisely a spending constraint step utility function.

Suppose we knew equilibrium prices for the given piecewise-linear utility functions. Using this information, let us obtain the corresponding spending constraint step utility functions and run the algorithm of this paper on this instance. Then, it is easy to see that the prices computed by the algorithm will be the same as the starting prices. Also, if the starting prices were not equilibrium prices for the given piecewise-linear instance, the computed prices will not be the same as the starting prices.

Now consider the following procedure. Start with an initial price vector so that the sum of the prices of all goods adds up to the total money possessed by buyers. Using these prices, convert the given piecewise-linear utility functions into spending constraint step utility functions and run the algorithm of the current paper on this instance to obtain a new price vector. Repeat until the price vector does not change, i.e., a fixed point is obtained. The question is does this procedure converge to a fixed point and if so how fast.

13 Acknowledgements

I wish to thank to John Ledyard for pointing out the work of Patinkin, Nikhil Devanur for suggesting that spending constraint utilities could be useful in the Adwords market, Aranyak Mehta for superbly playing the role of a critic and a sounding board when I was working out the toughest parts of the proof, and the anonymous referees of *MOR* for suggesting numerous improvements to the paper.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms and Applications.* Prentice Hall, N.J., 1993.

[2] K. Arrow and G. Debreu. Existence of an equilibrium for a competitive economy. *Econometrica*, 22:265–290, 1954.

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

[4] P. Bridel. Patinkin, Walras and the money-in-the-utility-function tradition. *European J. History of Economic Thought*, 9:2:268–292, 2002.

[5] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization.* John Wiley and Sons, Inc., N.Y., 1998.

[6] N. Devanur, C. H. Papadimitriou, A. Saberi, and V. V. Vazirani. Market equilibrium via a primal-dual-type algorithm. In *Proceedings of IEEE Annual Symposium on Foundations of Computer Science*, 2002. To appear in *JACM*. Journal version available at: *t*http://www-static.cc.gatech.edu/ vazirani/market.ps.

[7] N. Devanur and V. V. Vazirani. The spending constraint model for market equilibrium: Algorithmic, existence and uniqueness results. In *Proceedings of 36th STOC*, 2004.

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

[9] D. Gale. Piecewise linear exchange equilibrium. *Jounral of Mathematical Economics*, 4:81–86, 1977.

[10] N. Megiddo. A note on the complexity of p-matrix lcp and computing an equilibrium. IBM Research Report 6439. Available at: http://theory.stanford.edu/ megiddo/pdf/plcp.pdf, 1988.

[11] N. Megiddo and C.H. Papadimitriou. On total functions, existence theorems, and computational complexity. *Theoretical Computer Science*, 81:317–324, 1991.

[12] D. Patinkin. *Money, Interest, and Prices. An Integration of Monetary and Value Theory.* Harper and Row, N.Y., 1965.

[13] V. V. Vazirani. Nash bargaining via flexible budget markets. Submitted to *JACM*, 2008.