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 Nobel prize winning 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. 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 market equilibria is apparent.

Such an algorithmic 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 [2]. 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 – this 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. We also show that this case has other properties rivaling the traditional model, including 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.

In the sequel to this paper, [4] continue the study of spending constraint utilities. For the case that spending constraint functions are continuous and strictly decreasing, [4] 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.

[4] also give a natural way of defining spending constraint utilities in the exchange model of Arrow and Debreu [1]. 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 strictly concave utility functions for each good. 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. Indeed finding a good algorithm for these utility functions remains an important open problem in algorithmic game theory; see [6] 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, [3] 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.

It will be convenient to 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 on 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 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 his business together with his bid for each keyword. Each bid represents the amount he is willing to pay to Google if his ad is shown along with search results to the corresponding keyword and moreover the user clicks on the ad. The advertiser also specifies his daily budget – the maximum amount that Google can charge him 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?

Both, linear and concave utility functions are not suitable for this purpose. 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. On the other hand, although concave utility functions are expressive enough to capture very complex requirements of an advertiser, equilibrium prices and allocations for these utility functions is not known to be computable in polynomial time. 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.

1.3 Overview of algorithmic ideas

Spending constraint step utility functions generalize linear utility functions and our algorithm is obtained by generalizing the algorithm of [3]. Similar to [3], our algorithm is also based on the primal-dual schema, with allocations of goods playing the role of primal variables and prices playing the role of dual variables. As in [3], we start with very low prices on goods, and gradually raise prices until 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.

Let us outline the main ideas needed, beyond [3], for obtaining our algorithm. In the linear case, at any given prices, each buyer i has a set of most desirable goods. Any allocations made from this set of goods makes i equally happy; it is not *essential* to allocate any particular good from this set. Indeed, the algorithm of [3] exploits this freedom fully – it does not need to commit to any allocations as the prices are being raised; allocations are made only at the end, after equilibrium prices have been computed.

In our setting, 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 to buyer i and need to be deallocated (it is difficult to imagine an iterative scheme that avoids this possibility altogether).

The question is how do we arrange the algorithm so this backtracking does not end up taking exponential time? For this purpose, we give a systematic way of making allocations and deallocations so that the price of each good can keep increasing throughout. Consequently, the total price of all goods forms a convenient potential function for measuring progress of the algorithm.

2 Fisher's model and spending constraint utilities

Fisher's market model is the following. Let A be a set of divisible goods and B be a set of buyers, $|A| = n,\ |B| = n'$. 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, $b_j \in \mathbf{Q}^+$ of each good $j \in A$. For each buyer i and good j we are specified a function $h^i_j : \mathbf{R}^+ \rightarrow \mathbf{R}^+$ which gives the utility that i derives as a function of the amount of good j that she receives. Her overall utility for any allocation is additive 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, $h^i_j(x_{ij}) = u_{ij}x_{ij}$ where $u_{ij} \geq 0$ is a constant. Fisher had originally defined his model for the case that h^i_j is a strictly concave, differentiable function.

An easy way of describing *spending constraint step utility functions* is by contrasting with the case of piecewise-linear and concave functions. Suppose h^i_j is such a function. Let r^i_j be the derivative of h^i_j; this will be a decreasing step function. Observe that function r^i_j 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. 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 $j \in A$, 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 $g_j^i : [0, e(i)] \to \mathbf{R}^+$ 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 shown in [4].

Each buyer also has utility for the part of her money that she does not spend. For $i \in B$, let $f_0^i : [0, e(i)] \to \mathbf{R}^+$ 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 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 [2] 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 $\boldsymbol{p} = (p_1, \ldots, p_n)$ of all 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. Let 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.,

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

- 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., $\forall j \in A, b_j = 1$.
- 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 rate$(s)/p_j$. The bang per buck of segment $s \in \text{seg}(f_0^i)$ is simply rate(s). Sort all segments $s \in$ segments(i) by decreasing bang per buck, and partition by equality into classes: $Q_1, Q_2, \ldots$. For a class Q_l, define 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 is not viable, one has 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 1.2, that of choosing a utility function for advertisers in Google's adwords market. As argued in Section 1.2, both linear and 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". Thus the keyword "men's clog" is profitable only if its price per click is under \$2 and "women's clog" is profitable only if its price per click is under \$4 per click.

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 better keyword is at least twice as profitable as 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 profit on this keyword is at least \$2 per dollar spent on advertising** 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 j such that $p'_j < p_j$. Let

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

By assumption, $\theta < 1$. Let $S = \{j \in A \mid p'_j = \theta p_j\}$; 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 $\mathbf{p}'$. Let M and M' denote the amount of money spent on goods in S in these two bundles, respectively. Then, $M' \geq M$.*

Proof : Let Q_1 and Q_2 denote the set of segments in i's forced and flexible allocations, respectively, at prices $\boldsymbol{p}$. Similarly, let Q'_1 and Q'_2 denote the set of segments in i's forced and flexible allocations, respectively, at prices $\boldsymbol{p}'$.

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 $M' \geq M$. 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 proof is obvious.
2. All segments of Q_2 are in S. Now, by the fact given above, either $Q_2 \subset Q'_1$ or $Q_2 = Q'_2$ and $Q'_1 \subseteq Q_1$. 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 $P_1 \subset Q'_1$ or $P_1 = Q'_2$.

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

The algorithm iteratively raises prices until equilibrium prices are reached. On termination, the algorithm must end with the correct forced allocations for all buyers relative to the equilibrium prices. In addition, it must ensure that at intermediate points, the unique equilibrium price of any good is not exceeded. Two Invariants help ensure these two conditions.

We will denote by $\boldsymbol{p}$ the vector of current prices of all goods. At any intermediate point in the algorithm, certain segments are already allocated. By *allocating segment s*, $s \in \text{seg}(f^i_j), j \neq 0$, 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. In addition, at an intermediate point in the algorithm, some money would be returned to buyer i. Let returned(s), $s \in \text{seg}(f^i_0)$, 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 $s' \in \text{seg}(f^i_0)$ having 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 value of good j, $j \neq 0$, 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 $s \in \mathrm{seg}(f_0^i)$, returned($s$) is added to spent($i$). Also, define the *money left over* with buyer i, $m(i) = e(i) - \mathrm{spent}(i)$.

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 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, $\alpha(i)$, 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. Next, we define the *equality subgraph* $G = (A, B, E)$ on bipartition A, B and containing edges E. Corresponding to each buyer i and each segment $s \in Q^{(i)}$, E contains the edge (i, j), where good(s) $= j$. The capacity of this edge, $c_{ij} = \mathrm{value}(s)$.

Denote by $\boldsymbol{a}$, $\boldsymbol{s}$ and $\boldsymbol{m}$ the current allocations, amounts spent and left over money, i.e., (allocated(j), $j \in A$), (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 $S \subseteq A$, $\boldsymbol{m}(S)$ will denote $\sum_{j \in S} m(j)$.

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. Direct all edges of the equality subgraph, G, from A to B. Add a source vertex s, and directed edges (s, j), for each $j \in A$ and having capacity $p_j - \mathrm{allocated}(j)$. Add a sink vertex t, and directed edges (i, t), for each $i \in B$ and having capacity $m(i)$. Throughout the algorithm, we will maintain the following:

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

For $S \subseteq A$, define its *neighborhood in the equality subgraph* to be

$$\Gamma(S) = \{i \in B \mid \exists j \in S \text{ with } (i, j) \in G\}.$$

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

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

and define $\mathrm{bestT}(S)$ 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 one could find an even better set, T, with $T_2 \subset T \subset T_1$). Hence $\mathrm{bestT}(S)$ is unique. We can now give a characterization of Invariant 2 in terms of cuts in the network.

Lemma 3 *Network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ satisfies Invariant 2 iff*

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

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

For the reverse direction, assume Invariant 2 does not hold. Let $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ be a min-cut in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$, with $A_1, A_2 \subseteq A$ and $B_1, B_2 \subseteq B$. We may assume that $B_1 \subseteq \Gamma(A_1)$. Now, since $(s, A \cup B \cup t)$ is not a min-cut,

$$\text{best}(A_1) \leq \boldsymbol{m}(B_1) + c(A_1, \Gamma(A_1) - B_1) < \boldsymbol{p}(A_1) - \boldsymbol{a}(A_1).$$

$\triangle$

A nonempty set $S \subseteq A$ that satisfies the inequality in Lemma 3 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 4 *Assume that Invariant 2 holds. If $S_1 \subseteq A$ and $S_2 \subseteq A$ are two tight sets, then $S_1 \cup S_2$ is also a tight set.*

Proof : Corresponding to a set $S \subseteq A$ 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. Define this to be the *S-network.* Since Invariant 2 holds, $(s, A \cup B \cup t)$ is a min-cut in this network. 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, it would be possible to construct an S_1-network or an S_2-network which violates the residual path assertion stated above, hence contradicting tightness of the corresponding set. Hence, $S_1 \cup S_2$ is also a tight set. $\triangle$

Corollary 5 *If Invariant 2 holds, the maximal tight set is unique.*

6 Algorithm 1

For ease of exposition and comprehension, we will first present the algorithm assuming that buyers have no utility for money. In Section 11 we will remove this restriction. In this section, we will present an algorithm that terminates with equilibrium prices, and we will establish that equilibrium prices have polynomial sized descriptions. However, we do not know whether this algorithm runs in polynomial time. In Section 10, we will show that a suitable modification of this algorithm runs in polynomial time.

Observe that if S is a tight set, market clearing prices have been achieved for these goods – unless the equality subgraph undergoes change – since the prices of goods in S are just right to exactly exhaust the money of all buyers interested in these goods. Hence, the algorithm will stop raising prices of these goods (indeed, raising them will violate Invariant 2).

The algorithm partitions the equality subgraph $G = (A, B, E)$ into two: *frozen* and *active*, consisting of bipartitions (A_1, B_1) and (A_2, B_2), respectively (throughout this paper, A_1, A_2 will be subsets of A and B_1, B_2 will be subsets of B). A_1 is the maximal tight set of G, $B_1 = \Gamma(A_1)$, and the frozen subgraph satisfies $\boldsymbol{p}(A_1) - \boldsymbol{a}(A_1) = \boldsymbol{m}(B_1)$. The active subgraph satisfies

$$\forall S \subseteq A_2 : \ \boldsymbol{p}(S) - \boldsymbol{a}(S) < \text{best}(S)$$

and so prices of goods in A_2 can be raised without violating Invariant 2. The crucial job of partitioning the equality subgraph is performed by subroutine **freeze** (see Section 7), which also performs other related functions. As argued in Section 7, the frozen and active subgraphs are disconnected and hence decoupled.

In order to ensure Invariant 2 at the start of the algorithm, the following steps are executed:

- 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. Clearly, each buyer's current partition will be her first partition.
- Next, we have to ensure that each good j has an interested buyer, i.e., has an edge incident at it in the equality subgraph. Compute α_i for each buyer i at the prices fixed in the previous step and compute the equality subgraph. If good j has no edge incident, reduce its price to
$$p_j = \max_{i \in B} \max_{s \in \text{seg}(f_j^i)} \left\{ \frac{\text{rate}(s)}{\alpha_i} \right\}.$$

Next, partition the equality subgraph into frozen and active by calling subroutine **freeze** (see Section 7). Market clearing prices have not been reached for goods in the active subgraph and their prices need to be increased. We want to do this in such a way that the equality subgraph remains unchanged. Observe that if buyer i has equality edges to goods j and j' due to segment $s \in \text{seg}(f_j^i)$ and $s' \in \text{seg}(f_{j'}^i)$ then

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

This suggests increasing prices in such a way that the ratio of prices of any two goods is not affected, which in turn is accomplished as follows: Multiply the current price, p_j, of each good $j \in A_2$ by x. Initialize $x = 1$, and start raising x continuously.

As x is raised, one of the following three events could take place.

- **Event 1:** As prices increase, a subset of A_2 may go tight. If so, subroutine **freeze** is called to recompute the frozen and active subgraphs.
- **Event 2:** For buyers in B_2, goods in A_1 are becoming more and more desirable (since their prices are not changing, whereas prices of goods in A_2 are increasing). As a result, a segment $s \in \text{seg}(f_j^i)$, $i \in B_2$, $j \in A_1$ may enter into the current partition of buyer i, $Q^{(i)}$. When this happens, edge (i, j) is added to the equality subgraph. As a result, A_1 is not tight anymore, and therefore subroutine **freeze** is called to recompute the frozen and active subgraphs.
- **Event 3:** Suppose $i \in B_1$ has a segment $s \in \text{seg}(f_j^i)$ allocated to it, where $j \in A_2$. Since the price of j is increasing, at some point the bang per buck of this segment may equal α_i, i.e., segment s enters i's current partition. When this happens, we will *deallocate* segment s, i.e., subtract value(s) from allocated(j) and from spent(i) and add edge (i, j) to the active subgraph. Since $m(i)$ increases, A_1 is not tight anymore, and therefore subroutine **freeze** is called to recompute the frozen and active subgraphs.

Remark 6 *Because of forced allocations, there may be a good j in the frozen subgraph that has no equality subgraph edges incident at it. If so, by Invariant 2 it must be the case that $p_j = \text{allocated}(j)$.*

When all goods enter the frozen subgraph, the algorithm terminates and outputs the current prices of goods – these will be equilibrium prices. Computation of equilibrium allocations is now straightforward an uses the following fact: if the equilibrium price of good j is p_j then the amount of good j corresponding to a segment s is $\text{value}(s)/p_j$.

Algorithm 1 is summarized below. It 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, and minimum of these is the event that happens first. For Events 2 and 3, the computation is straightforward. Let x^* be the value of x at which Event 1 happens. We give a procedure for computing x^* in Section 8.

```
Initialization:
$\forall j \in A, p_j \leftarrow 1/n$;
$\forall i \in B, \alpha_i \leftarrow \text{rate}(s)/\text{good}(s),\ s \in Q^{(i)}$;
Compute equality subgraph $G$;
$\forall j \in A$ if $deg_G(j) = 0$ then $p_j \leftarrow \max_{i\in B} \max_{s\in \text{seg}(f_j^i)} \left\{\frac{\text{rate}(s)}{\alpha_i}\right\}$;
Recompute $G$;
$(A_1, B_1) \leftarrow (\emptyset, \emptyset)$ (The frozen subgraph);
$(A_2, B_2) \leftarrow (A, B)$ (The active subgraph);
while $A_2 \neq \emptyset$ do
    $x \leftarrow 1$;
    Define $\forall j \in A_2$, price of $j$ to be $p_j x$;
    Raise $x$ continuously until one of three events happens:
    if $S \subseteq A_2$ goes tight then
        Call freeze;
    if segment, $s$, corresponding to $i \in B_2$ , $j \in A_1$ enters $Q^{(i)}$, then
        Add $(i, j)$ to $G$ with $c_{ij} = \text{value}(s)$;
        Call freeze;
    if allocated segment, $s$, corresponding to $i \in B_1$ , $j \in A_2$ enters $Q^{(i)}$, then
        Deallocate $s$;
        Add $(i, j)$ to $G$ with $c_{ij} = \text{value}(s)$;
        Call freeze;
```

Algorithm 1:

7 Subroutine freeze and forced allocations

All forced allocations of Algorithm 1 are made by subroutine **freeze** which operates as follows. Via max-flow, find a min-cut in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ that maximizes the number of vertices in the s side (there is a unique such maximal min-cut). Let it be $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ [1]. Clearly, $B_1 = \text{bestT}(A_1)$.

[1]Throughout this paper, we will assume that $A_1, A_2 \subseteq A$ and $B_1, B_2 \subseteq B$.

Corresponding to each edge connecting A_1 to $\Gamma(A_1) - B_1$, allocate goods. These are the forced allocations. Consider good $j \in A_1$. Observe that if the total forced allocations of good j that need to be made exceed $p_j - \text{allocated}(j)$ then moving j into A_2 would result in a better min-cut in N. Similarly, consider buyer $i \in B_2$. Again, if the total forced allocations that need to be made to i exceed her left over money, $m(i)$, then moving i into B_1 would result in a better min-cut in N. Therefore, we are never in danger of making $\text{allocated}(j) > p_j$ or exceeding the left over money of a buyer while making the required forced allocations.

Remark 7 *Observe that under the spending constraint step utility functions, a segment $s \in \text{seg}(f^i_j)$ represents $s)$ 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. This way of viewing segments, i.e., as representing a certain value rather than quantity of good, is much more convenient to deal with via an algorithm that iteratively adjusts prices. Under usual piecewise linear utility functions, each piece would have represented a certain quantity of good. Making the corresponding forced allocation would have led to the cumbersome issue of keeping track of its value, and consequently the left over money of the buyer, as the price of this good changes.*

As a result of these allocations, there may be buyers in B_2 that do not have any equality edges incident at them (however, by the fact that $B_1 = \text{bestT}(A_1)$, they must have money left over). For each such buyer i, compute her partitions relative to current prices and include edges corresponding to the first unallocated partition. If none of these edges is incident at a good in A_1, subroutine **freeze** returns to the main algorithm. Otherwise, the following event is triggered.

- **Event 4:** Since new edges have been introduced between B_2 and A_1, the current set A_1 is not tight anymore. Therefore, subroutine **freeze** is called to recompute the frozen and active subgraphs.

Before returning, subroutine **freeze** partitions the equality subgraph into two: *frozen* and *active*. The frozen subgraph consists of the bipartition (A_1, B_1) and the active subgraph consists of (A_2, B_2).

Observe that buyers in B_1 may desire goods in A_2. By Lemma 8, the prices of these goods can be raised without violating Invariant 2. As soon as this happens, buyers in B_1 who have equality edges to goods in A_2 will not be interested in these goods anymore, and such edges can be dropped. The frozen and active graphs are hence decoupled. After these changes, the following holds:

Lemma 8 *The active and frozen subgraphs satisfy Invariant 2. Furthermore, the active subgraph satisfies:*

$$\forall S \subseteq A_2 : \ \boldsymbol{p}(S) - \boldsymbol{a}(S) < \text{best}(S),$$

and the frozen subgraph satisfies:

$$\boldsymbol{p}(A_1) - \boldsymbol{a}(A_1) = \boldsymbol{m}(B_1).$$

Proof : The min-cut found by **freeze** has the same capacity as $(s, A \cup B \cup t)$, which is also a min-cut, since Invariant 2 holds. Therefore, $\boldsymbol{p}(A_1) - \boldsymbol{a}(A_1) = \text{best}(A_1)$. Therefore after forced allocations are made, we have: $\boldsymbol{p}(A_1) - \boldsymbol{a}(A_1) = \boldsymbol{m}(B_1)$.

Consider the situation after **freeze** has made all required forced allocations and is ready to return. Compute max-flow in network $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ at this point. Now, any flow through $S \subseteq A_1$ must leave via B_1. Therefore the frozen subgraph satisfies Invariant 2. Any edge from A_2 to B_1 cannot carry any flow, since this edge goes from the t-side to the s-side of the min-cut. Therefore any flow through $S \subseteq A_2$ must leave via B_2. Hence the active subgraph also satisfies Invariant 2. Now, if there is a set $S \subseteq A_2$ such that $\boldsymbol{p}(S) - \boldsymbol{a}(S) = \text{best}(S)$, then moving it into A_1, together with appropriate changes to B_1 and B_2, still yields a min-cut. This contradicts maximality of the min-cut found. Hence, the active subgraph satisfies the above-stated inequality. $\triangle$

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

For simplicity of notation, assume that the active subgraph is (A, B). Throughout this section, $\boldsymbol{p}$ will denote prices at the beginning of the current phase, i.e., at $x = 1$. We first show how to compute x^*, the value of x at which Event 2 occurs, i.e., a new set goes tight. Let $S^* \subseteq A$ denote the tight set. In $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$, replace the capacities of edges (s, j), $j \in A$, by $p_j \cdot x - \text{allocated}(j)$ to obtain the parametric network $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. By Invariant 2, at $x = 1$, $(s, A \cup B \cup t)$ is a min-cut in $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$.

Lemma 9 *The smallest value of x at which a new min-cut appears in $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ is given by*

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

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 $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$, and suppose it is $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$. Since $(s, A \cup B \cup t)$ is also a min-cut in $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$,

$$\boldsymbol{p}(A_1) \cdot \beta - \boldsymbol{a}(A_1) = \text{best}(A_1).$$

On the other hand, for any other set $S \subseteq A$,

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

The lemma follows. $\triangle$

Lemma 10 *The following hold:*

- *If $x \leq x^*$, then $(s, A \cup B \cup t)$ is a min-cut in $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$.*
- *If $x > x^*$, then for any min-cut $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ in $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$, $S^* \subseteq A_1$.*

Proof : By definition of x^*, if $x \leq x^*$, $\forall S \subseteq A : \ \boldsymbol{p}(S) \cdot x - \boldsymbol{a}(S) \leq \text{best}(S)$. Therefore, by Lemma 3, Invariant 2 holds and hence $(s, A \cup B \cup t)$ is a min-cut in $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$.

Next, suppose that $x > x^*$, and consider a min-cut $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ in $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. First observe that $S^* \subseteq A_2$ 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 A_1, together with appropriate changes to B_1 and B_2.

Let $S^* \cap A_1 = S_1$, $S^* \cap A_2 = S_2$, 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:

$$\text{best}'(S_2) = \min_{T \subseteq \Gamma(S_2) - B_1} \{\boldsymbol{m}(T) + c(S_2, \Gamma(S_2) - B_1 - T)\}.$$

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 A_1 (together with other changes to B_1 and B_2 according to $\text{best}'(S_2)$). 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 $S^* \subseteq A_1$. $\triangle$

For $i \in B$, denote the sum of capacities of edges incident at i in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ by $c(i)$. Define $m'(i) = \min\{m(i), c(i)\}$, and $\boldsymbol{m}'$ to be the vector consisting of $m'(i), i \in B$. Observe that replacing $\boldsymbol{m}$ by $\boldsymbol{m}'$ in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ does not change the min-cut or its capacity. Define $N''(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ to be the network obtained by replacing $\boldsymbol{m}$ by $\boldsymbol{m}'$ in $N'(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. The reason for working with $\boldsymbol{m}'$ is that the cut $(s \cup A \cup B_1, B_2 \cup t)$ has the same capacity as the cut $(s \cup A \cup B, t)$. This property will be used critically in the next lemma.

Lemma 11 *Set $x = (\boldsymbol{m}'(B) + \boldsymbol{a}(A))/\boldsymbol{p}(A)$ and find the minimal min-cut in $N''(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ (i.e., the unique min-cut minimizing the s side). Let it be $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$. If $A_1 = B_1 = \emptyset$ then $x = x^*$ and $S^* = A$. Otherwise, $x > x^*$ and A_1 is a proper subset of A.*

Proof : Clearly, $x \geq x^*$. If the min-cut is $(s, A \cup B \cup t)$ then by Lemma 10, $x = x^*$. For the chosen value of x, $x \cdot \boldsymbol{p}(A) - \boldsymbol{a}(A) = \boldsymbol{m}'(B)$ and by the property stated above, $\text{best}(A) = \boldsymbol{m}'(B)$. Therefore $\text{best}(A) = x^* \cdot \boldsymbol{p}(A) - \boldsymbol{a}(A)$, and hence $S^* = A$.

If $(s, A \cup B \cup t)$ is not a min-cut, then by Lemma 10, $x > x^*$. Suppose $A_1 = A$ and the min-cut is $(s \cup A \cup B_1, B_2 \cup t)$. By the property stated above, the capacity of this cut is $\boldsymbol{m}'(B)$. For the chosen value of x, the capacity of $(s, A \cup B \cup t)$ is also the same, thereby contradicting the fact that it is not a min-cut. Hence A_1 is a proper subset of A. $\triangle$

Lemma 12 *x^* and S^* can be found using n max-flow computations.*

Proof : Let $x = (\boldsymbol{m}'(B) + \boldsymbol{a}(A))/\boldsymbol{p}(A)$ and compute a min-cut in $N''(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. If $(s, A \cup B \cup t)$ is a min-cut in $N''(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ then by Lemma 11, $x^* = x$ and $S^* = A$. Otherwise, $x > x^*$. Let $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ be a min-cut in $N''(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. By Lemmas 10 and 11, $S^* \subseteq A_1 \subset A$. Therefore, it is sufficient to recurse on the network restricted to $(A_1, \Gamma(A_1))$ (of course $\boldsymbol{m}'$ will have to be recomputed for this restricted network). $\triangle$

9 Termination with market clearing prices

Observe that despite the return policy, the algorithm monotonically keeps raising prices of goods, and this provides us with a natural measure of progress – the difference between total money possessed by buyers (after taking into consideration money returned) and the sum of the prices of all goods. When this difference becomes zero, all goods must be frozen, and the algorithm terminates. If Invariants 1 and 2 hold, terminating prices are market clearing.

Let M denote the total amount of money possessed by the buyers, U denote the largest rate of a segment. Let $\Delta = nU^n$. Let us partition the run of the algorithm into *phases* – each phase ends when a new set goes tight, i.e., Event 1 occurs. Partition each phase into *iterations* – each iteration ends when Event 2, 3 or 4 occurs.

Lemma 13 *Algorithm 1 maintains Invariants 1 and 2 throughout.*

Proof : Algorithm 1 maintains Invariant 1 because it responds correctly to changes to Q_{t_i}, the current partition of buyer i, for each i. The following is the list of possible changes to Q_{t_i}, the current partition of buyer i, assuming that for segment s, good$(s) \neq 0$, i.e., it does not represent money. In each case, we argue that Algorithm 1 responds correctly to these changes and hence maintains Invariant 1.

- If segment s moves from Q_{t_i} to Q_{t_i-1}, Algorithm 1 allocates segment s.
- If segment s moves from Q_{t_i-1} to Q_{t_i}, Algorithm 1 deallocates segment s (i must be in the frozen subgraph and good(s) must be in the active subgraph).
- If segment s moves from Q_{t_i} to Q_{t_i+1}, Algorithm 1 removes the edge corresponding to segment s from the equality subgraph (this happens when i moves to the frozen subgraph but good(s) remains in the active subgraph).
- If segment s moves from Q_{t_i+1} to Q_{t_i}, Algorithm 1 adds an edge corresponding to segment s (i must be in the active subgraph and good(s) must be in the frozen subgraph).
- If Q_{t_i} becomes empty (because all its segments moved or got allocated), Algorithm 1 defines Q_{t_i+1} to be the current partition for i.

Observe that if Invariant 2 holds before Events 2 and 3 then it holds after these events as well: the addition of edge (i, j), $i \in B_2, j \in A_1$, or the deallocation of a segment can only increase best(S) for any set $S \subseteq A$. By Lemma 8, the steps taken in subroutine **freeze** also maintain Invariant 2. Finally, Event 1 also cannot affect Invariant 2, since as soon as it is triggered, a set is frozen. $\triangle$

The proof of the following lemma is the same as that of Lemma 8 in [3].

Lemma 14 *At the termination of a phase, the prices of goods in the tight set are rational numbers with denominators* $\leq \Delta$.

Lemma 15 *A phase consists of at most* n *iterations.*

Proof : Observe that within Events 2, 3 and 4, the addition of edge (i, j), $i \in B_2, j \in A_1$, or the deallocation of a segment can only increase best(S) for any set $S \subseteq A$, and therefore cannot result in a new tight set. On the other hand, after each of these steps, $\boldsymbol{p}(A_1) - \boldsymbol{a}(A_1) < \text{best}(A_1)$. Therefore when **freeze** is called, at least one good gets unfrozen. The lemma follows. $\triangle$

Lemma 16 *Algorithm 1 executes at most $M\Delta^2$ phases, where M is the total money of all buyers.*

Proof : Consider a freezing due to Event 1, and assume that (S, T), with $S \subseteq A_2, T \subseteq B_2$, is the new subgraph that gets frozen. If $T = \emptyset$, then for each good $j \in S$, $\boldsymbol{p}(j) - \boldsymbol{a}(j)$ must equal the sum of capacities of edges incident at j, which is integral. Therefore $\boldsymbol{p}(j)$ is integral. If $T \neq \emptyset$, then via the argument presented in Lemma 8 in [3], one can show that for each good $j \in S$, the denominator of the current price of j is $\leq \Delta$.

Now, using the argument presented in Lemma 11 in [3], one can show that k phases must lead to an increase in the total price of all goods by at least k/Δ^2. The lemma follows. $\triangle$

Each iteration requires a computation of x^*, which requires n max-flow computations by Lemma 12. Using Lemmas 15 and 16 we get:

Theorem 17 *Algorithm 1 terminates with equilibrium prices and allocations, and executes at most $O(M\Delta^2 n^2)$ max-flow computations.*

10 Establishing polynomial running time

Algorithm 1 makes steady progress toward finding the equilibrium – it never lowers the price of any good and therefore the flow from s to t increases monotonically. However, we have not been able to show that it terminates in polynomial time. In this section we will modify Algorithm 1 appropriately to give an algorithm for which we can establish polynomial time termination. For this purpose, we will extend the notion of balanced flows from [3] to our more involved setting which is caused by the fact that in our network N the edges between A and B have finite capacities. Our generalization turns out to be a natural one.

In Algorithm 1 we were measuring progress by the flow from s to t, or equivalently, the total surplus money of all buyers. In the new algorithm we will measure progress using the l_2 norm of the vector of surplus money of the buyers. This potential function 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 simply N. We will assume that network N satisfies Invariant 2, i.e., $(\{s\}, A \cup B \cup \{t\})$ 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 $\|\boldsymbol{\gamma}(N, f)\|$. It is easy to see that a balanced flow must be a max-flow in N. Furthermore, using the same proof as Lemma 14 in [3], one can see that all balanced flows in N have the same surplus vector, which we will denote by $\boldsymbol{\gamma}(N)$. In addition, using the same proof as Theorem 15 in [3], one can show that a maximum flow f in N is balanced iff it satisfies:

Property 1: If $\boldsymbol{\gamma}_j(N, f) < \boldsymbol{\gamma}_i(N, f)$ then there is no path from node j to node i in $R(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s}, f) - \{s, t\}$.

10.1 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 $\{s\} \cup A \cup B \cup \{t\}$ in polynomial time. As stated above, we will assume that this network satisfies Invariant 2, i.e., $(\{s\}, A \cup B \cup \{t\})$ is a min-cut in N.

Continuously reduce the capacities of all edges that go from B to t, other than those edges whose capacity becomes zero, until the capacity of the cut $(\{s\} \cup A \cup B, \{t\})$ becomes the same as the capacity of the cut $(\{s\}, A \cup B \cup \{t\})$. 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 $t \in T$.

Case 1: If $T = \{t\}$ then find a max-flow in N' and output it – this will be a balanced flow in N.

Case 2: Otherwise, let N_1 and N_2 be the subnetworks of N induced by $S \cup \{t\}$ and $T \cup \{s\}$, respectively. Let A_1 and B_1 be the subsets of A and B, respectively, induced by N_1. Similarly, let A_2 and B_2 be the subsets of A and B, respectively, induced by N_2. Let F be the set of edges that go from A_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, the capacities of some edges from s to A_1 and B_2 to t will be used up. Update the capacities of the corresponding 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.

Remark 18 *Some of the edges saturated by h will lead to forced allocations in Algorithm 2.*

Lemma 19 *f is a max-flow in N.*

Proof : In the first case, i.e., $T = \{t\}$ in the min-cut in N', the algorithm outputs a max-flow in N'. This flow must saturate the cut $(\{s\} \cup A \cup B, \{t\})$. However, since the capacity of this cut in N' is the same as the capacity of the cut $(\{s\}, A \cup B \cup \{t\})$, by Invariant 2, this is also a max-flow in N.

Next let us consider the second case. 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. By Invariant 2, a max-flow in N must saturate all edges from s to A. Let us show that f accomplishes this.

Since edges F are in the min-cut in N', they must be saturated in f'. Therefore, there must be a max-flow in N, say g, that saturates all edges in F. Now observe that restricting g to M_1 is a max-flow in M_1 that saturates all edges from s to A_1 in M_1. Hence f_1 also does so. Next restrict f' to M_2. This must be a feasible flow in M_2 and it must also be a max-flow since it saturates all edges from s to A_2. Hence f_2 also must do so. $\triangle$

Lemma 20 *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 19, 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 these two networks.

Let R be the residual graph of N w.r.t. flow f; we only need to 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 A_1 to B_2 in R, and therefore there are no paths from $j \in B_1$ to $i \in B_2$. There may however 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 min-cut found in N', all nodes in B_2 have surplus capacity > 0 w.r.t. flow f' in N' (all nodes having surplus zero must be in B_1). Therefore, the same amount, say X, 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 $\gamma_i(f) > X$ for each $i \in B_2$. A similar proof shows that $\gamma_i(f) < X$ for each $i \in B_1$, thereby establishing Property 1.

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 A_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 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 hence in f.

Let $c(K)$ denote the sum of 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 can use edges of F' and the rest, i.e., at least $c(K) - c(F')$ 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. By the argument given above,

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

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

$$c(L) = c'(L) + |L|X.$$

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

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

Now, since all L edges in E_L have the same surplus, each has surplus $> X$. The lemma follows. $\triangle$

Theorem 21 *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 20, the flow output by the algorithm is a balanced flow in N. $\triangle$

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 Algorithm 2.

Lemma 22 *The surplus vectors of balanced flows in N and $N_{F'}$ are the same, i.e., $\gamma(N) = \gamma(N_{F'})$.*

Proof : The lemma follows from the fact that by Lemma 20 f is a balanced flow in N in which all edges of F' are saturated. △

10.2 The modified algorithm

In this section we will give the modified algorithm, Algorithm 2, which uses balanced flows to compute the equilibrium. The initialization step, which ensures that Invariant 2 is satisfied, is identical to that in Algorithm 1. The run of the algorithm is again partitioned into *phases*, each phase ends with a new set going tight. Each phase is partitioned into iterations which are defined below.

A phase starts with computation of a balanced flow, say f, in the current network, $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$. If the algorithm of Section 10.1 for finding a balanced flow terminates in Case 1, then the current prices and allocations are equilibrium prices and allocations and the algorithm halts. 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 **.

Step ** : 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. Let J' be the set of goods that are have edges to I in N and let $J = J' - Q$. Corresponding to each edge in $N(\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{s})$ which goes from Q to I, allocate goods; these are the *forced allocations.* Observe that because of Property 1, these edges must be saturated in f. If as a result of these forced allocations there are buyers $i \in I$ that have no more edges incident at them, go to Event 4.

The network induced by $I \cup J$ is the *active subgraph* and that induced by $P \cup Q$ is the *tight 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 three events happens. Observe that as soon as $x > 1$, buyers in $B - I$ are no longer interested in goods in J and all such edges can be dropped from the equality subgraph and N.

- **Event 1:** If a subset $S \subseteq J$ goes tight, the current phase terminates an the algorithm starts with the next phase.
- **Event 2:** If an edge (i, j), with $i \in I$ and $j \in A - J$ enters the equality subgraph, add directed edge (j, i) to network N and go to Step **.
- **Event 3:** A segment s enters buyer i's current partition, where $i \in B - I$, $s \in \text{seg}(f_j^i)$ 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. In this case, 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 **.

- **Event 4:** W.r.t. current prices, compute the current partition for each buyer $i \in I$ having no equality subgraph edges. Add all corresponding edges to N and go to Step **.

Within a phase, we will call each occurrence of Events 1, 2, 3 and 4 an *iteration.* Let Z denote the total number of segments in all specified utility functions.

Lemma 23 *The total number of iterations in a phase is bounded by $Z + n$.*

Proof : After an iteration due to Event 2, at least one new buyer must move into the active subgraph. Since there is at least one buyer in the active subgraph at the start of a phase, the total number of iterations in a phase due to Event 2 i at most $n - 1$.

Observe that once a buyer enters the active subgraph, she must remain in it for the rest of the phase, and the same is true of goods as well. Suppose buyer i is not in the active subgraph. Then, any goods due to which it is deallocated segments must be in the active subgraph (their price must be increasing). Once i enters the active subgraph, these segments will be inferior to segments in her current partition, and since the price of these goods is increasing, these segments will not enter into her current partition in the present phase and hence will not be allocated. Therefore, the total number of times a segment is allocated and deallocated in a phase is at most once, leading to a bound of Z on the total number of iterations due to Events 3 and 4. Finally, the last iteration in each phase is due to Event 1. The lemma follows. $\triangle$

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, $0 \le t \le k$.

Lemma 24 *f_t is a feasible flow in N_{t+1}, for $0 \le t < k$.*

Proof : First observe that by Lemma 22, the forced allocations made do not change the surplus vector. Each of the following actions can only lead to a network that supports an augmented max-flow, hence giving the proof.

- Raising the prices of goods in J.
- Adding an edge, as required in Event 2.
- Deallocating a segment, as required in Event 3.
- Adding edges corresponding to current partitions of buyers in I, as required in Event 4.

$\triangle$

Corollary 25 $\|\gamma(N_t)\|$ *is monotonically decreasing with t.*

Let δ_t denote the minimum surplus of a buyer in the active subgraph in network N_t, for $0 \le t < k$; clearly, $\delta_0 = \delta$.

Lemma 26 *If $\delta_{t+1} < \delta_t$ then $\|\gamma(N_t)\|^2 - \|\gamma(N_{t+1})\|^2 \ge (\delta_t - \delta_{t+1})^2$, for $0 \le t < k$.*

Proof : By the same proof as that of Lemma 19 in [3], if $\delta_{t+1} < \delta_t$ then there is a buyer i whose surplus drops by $\delta_t - \delta_{t+1}$ in going from f_t to f_{t+1}. The statement of Lemma 17 in [3] holds for our setting as well; moreover the proof is the same. Now, since f_t is a feasible flow in N_{t+1}, we get the desired conclusion. △

Lemma 27

$$\|\gamma(N_0)\|^2 - \|\gamma(N_k)\|^2 \geq \frac{\delta^2}{(Z+n)}.$$

Proof : The left hand side can be written as a telescoping sum in which each term is of the form $\|\gamma(N_t)\|^2 - \|\gamma(N_{t+1})\|^2$. By Corollary 25, each of these terms is positive.

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 26 and the fact that $\delta_0 = \delta$ and $\delta_k = 0$, we get that

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

By Lemma 23, $k \leq Z + n$. The lemma follows. △

Lemma 28 *In a phase, the square of the l_2 norm of the surplus vector drops by a factor of*

$$\left(1 - \frac{1}{n(Z+n)}\right).$$

$$\|\gamma(N_0)\|^2 - \|\gamma(N_k)\|^2 \geq \frac{\delta^2}{(Z+n)}.$$

Proof : Lemma 27 and the fact that $\|\gamma(N_0)\|^2 \leq n\delta^2$ gives us

$$\|\gamma(N_k)\|^2 \leq \|\gamma(N_0)\|^2 \left(1 - \frac{1}{n(Z+n)}\right).$$

The lemma follows. △

Theorem 29 *Algorithm 2 finds equilibrium prices and allocations for spending constraint step utility functions in Fisher's model using*

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

max-flow computations.

Proof : By Lemma 28, the square of the surplus vector drops by a factor of half after $O(n(Z+n))$ phases. At the start of the algorithm, the square of the surplus vector is at most M^2. Once its value drops below $1/\Delta^4$, the algorithm requires only one more phase to compute equilibrium prices. This follows from Lemma 14 (which is identical to Lemma 8 in [3]) and Lemma 10 of [3] which applies to Algorithm 2 as well. Therefore the number of phases is

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

Since each phase By Lemma 23 each phase consists of $Z+n$ iterations and by Lemma 12 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 buyer i. We show how to modify Algorithm 2 to incorporate this; the changes needed to Algorithm 1 are similar.

We will say that buyer i has a *partially returned segment* $s \in \text{seg}(f_0^i)$ if $0 < \text{returned}(s) < \text{value}(s)$. This happens if i moves to the frozen subgraph before value(s) money corresponding to segment s could be fully returned. If so, when i returns to the active subgraph, the algorithm attempts to return the rest of value(s) to i. The following event is executed as the first event in each iteration.

Event 0: This event happens if I contains buyer i with $\text{rate}(s) = \alpha_i$ for $s \in seg(f_0^i)$, where $\text{returned}(s) < \text{value}(s)$. (This event happens in one of two ways: First, buyer i with a partially returned segment moved to I as a result of updating I. Second, as prices increase, the bang per buck of i decreased to the point where she is equally happy leaving with money corresponding to segment s unspent. In either case, the algorithm must return money corresponding to s before it can raise prices of goods.) The algorithm starts raising returned(s) continuously until one of two events happens:

- **Event 0(a):** Observe that for a set $S \subseteq A_2$ such that $i \in \text{bestT}(S)$, best(S) is decreasing as returned(s) is raised (since $m(i)$ is decreasing). As a result such a set may go tight. When this happens, go to Step **.
- **Event 0(b):** $\text{returned}(s) = \text{value}(s)$. In this case, money corresponding to segment s has been fully returned. Go back to raising the prices of goods in J.

In case of Event 0(a), let y^* be the value of returned(s) at which Event 0(a) occurs. Next, we show in case of Event 0 how to determine whether Event 0(a) or Event 0(b) occurs, and in the former case, how to compute y^*. Compute prices of all goods for the current value of x, and let them be denoted by $\boldsymbol{p'}$. Let $\boldsymbol{a}$ denote all forced allocations made so far. Compute the money returned to buyers; for i, assume that segment s is fully returned. Let $\boldsymbol{s'}$ denote the vector of money spent. Construct network $N(\boldsymbol{p'}, \boldsymbol{a}, \boldsymbol{s'})$ and find a maximal min-cut in it. If $(s, A \cup B \cup t)$ is the only min-cut in it, 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 $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$, with $A_1 \neq \emptyset$. If so, Event 0(a) occurs. Clearly, the procedure stated above uses one max-flow computation.

Lemma 30 *If Event 0(a) occurs,*

$$y^* = \text{value}(s) - (\boldsymbol{p'}(A_1) - \boldsymbol{a}(A_1) - (\boldsymbol{m}(B_1) + c(A_1; \Gamma(A_1) - B_1))).$$

Proof : Call the above-stated situation, with the entire money corresponding to segment s returned to i, as Situation 1. Next, assume that the money returned to i, corresponding to segment s, is precisely y^*. Call this Situation 2, and let C be the maximal min-cut in this situation. Observe that i will be on the s-side of this cut. Therefore the capacity of C in Situation 1 will be smaller by exactly value$(s) - y^*$. On the other hand, the difference in capacity of any other cut in the two situations is at most value$(s) - y^*$. Therefore, C is a min-cut in Situation 1 as well, and the maximal min-cut computed in Situation 1 is precisely C.

Since Invariant 2 holds, $(s, A \cup B \cup t)$ will also be a min-cut in Situation 2. Therefore the drop in the capacity of $(s \cup A_1 \cup B_1, A_2 \cup B_2 \cup t)$ in the two situations is precisely $\boldsymbol{p}'(A_1) - \boldsymbol{a}(A_1) - (\boldsymbol{m}(B_1) + c(A_1; \Gamma(A_1) - B_1))$. This justifies the expression given above for computing y^*. $\triangle$

We prove below a lemma that is analogous to Lemma 14 for the enhanced model.

Lemma 31 *If Event 0(a) occurs,* returned(s) *is a rational number with denominator* $\leq \Delta$.

Proof : Let (S, T), with $S \subseteq I$ and $T \subseteq J$, be the new subgraph that gets frozen as a result of Event 0(a). Then, $i \in T$ and $\boldsymbol{p}(S) - \boldsymbol{a}(S) = \boldsymbol{m}(T) = \text{best}(S)$, where $\boldsymbol{m}(T)$ is computed assuming returned(s) money corresponding to segment s is returned. In (S, T) pick the connected component containing i. Let this be (S', T'). Now, $\boldsymbol{p}(S') - \boldsymbol{a}(S') = \boldsymbol{m}(T')$ because otherwise some other connected component of (S, T) will violate Invariant 2. Pick a spanning tree, τ, in (S', T').

Observe that the bang per buck of segment s is the same as the bang per buck of goods that i has edges to. Therefore, if $(i, j) \in \tau$,

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

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', then

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

Observe that the denominators involve rates of at most n segments. Therefore the denominator of $\boldsymbol{p}(S')$ is bounded by Δ. Also, because of the order in which events are executed, at most one buyer can have a segment that is not fully returned. Therefore the money left over with all buyers, other than i, in T' is integral. Also, $\boldsymbol{a}(S')$ integral. Hence, returned(s) is a rational number with denominator $\leq \Delta$. $\triangle$

Observe that if Event 0(b) occurs while returning money corresponding to segment s, then this segment will never be considered again. 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 29 holds for the enhanced model as well, as long as Z is taken to be the total number of segments in all specified utility functions, including those for money.

12 Open Problems

The remarkable convex program given by Eisenberg and Gale [5] 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 existence of equilibrium under certain mild conditions, 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 question, since in our model utilities of buyers are not only a function of allocations but also prices of goods, whereas in the Eisenberg-Gale program, prices are Lagrangian variables corresponding to packing constraints occurring in the program.

The fact that in our model, utilities of buyers are not only a function of allocations but also prices of goods can be considered a shortcoming of the model. In particular, the same bundle of goods may give a buyer a different utility by changing the relative prices of goods. One way to resolve this is to assume that buyers have a notion of “fair prices” of goods, and their utility is maximized not only when they get a desirable bundle of goods but also when the relative prices of goods is “fair”.

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 check if it generalizes to settling the latter question 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 do not satisfy weak gross substitutability. Our algorithm for spending constraint step utility functions may help finesse this difficulty via the following scheme given in [4]. 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

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

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 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 David Gale, Herve Moulin, Christos Papadimitriou, Eva Tardos and Pete Veinott for valuable discussions and comments, with special thanks to Nikhil Devanur, Aranyak Mehta and the anonymous Referee 1.

References

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

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

[3] N. Devanur, C. H. Papadimitriou, A. Saberi, and V. V. Vazirani. Market equilibrium via a primal-dual-type algorithm. *Submitted to JACM*, 2004. Available at: http://www-static.cc.gatech.edu/fac/Vijay.Vazirani/.

[4] N. Devanur and V. V. Vazirani. Extensions of the spending constraint model: Existence and uniqueness of equilibria. *In preparation.*

[5] E. Eisenberg and D. Gale. Consensus of subjective probabilities: The pari-mutuel method. *Annals Of Mathematical Statistics*, 30:165–168, 1959.

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