

# 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. The latter include weak gross substitutability, and the uniqueness and continuity of equilibrium prices and utilities.

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

**MSC 2000 Classification Codes:** Primary: 68Q25; Secondary: 91B16, 91B26, 91B32.

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

**History:** Received: December 25, 2008; Revised: December 22, 2009, March 2, 2010, March 8, 2010.

---

\*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 [13] (see also the subsequent paper [14]), 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, the 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 the 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 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 with polynomial descriptions if all input parameters are rational. In addition, as shown in [19], this case satisfies continuity of equilibrium prices and utilities.

The sequel to this paper, [10], continues the study of spending constraint utilities. For the case that spending constraint functions are continuous and strictly decreasing, [10] establishes the existence (using Brauer’s fixed point theorem) and the 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.

[10] also gives 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, [10] builds on our polynomial time algorithm for Fisher’s model to obtain FPTAS’s. Furthermore, for continuous and strictly decreasing spending constraint functions, [10] shows the existence of equilibrium prices using the Kakutani fixed point theorem (Brauer’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 (see [12] for an early work giving an algorithm for the case of two traders in the exchange model). The long-standing open problem of determining the complexity of finding an equilibrium under these utilities was resolved very recently – it was shown to be PPAD-complete for both Fisher and Arrow-Debreu models. First, [5] established PPAD-hardness for the Arrow-Debreu model. Next, [6, 20] concurrently showed PPAD-hardness for the Fisher model. In addition, [20] proved membership in PPAD for both market models.

These results, which establish evidence of intractability, and the fact that markets in the West, based on Adam Smith’s free market principle, seem to do a good job of finding prices that maintain parity between supply and demand, have prompted the author to ask whether we have failed to capture some essential elements of real markets in our models. In this context, our paper is demonstrating that it is possible to design interesting, natural utility functions for which an equilibrium can be computed efficiently.

Linear utility functions do satisfy weak gross substitutability and by exploiting this property, [9] 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 [16]; 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. Indeed Nisan et. al., faced with the problem of designing an auction system for Google for TV ads, converged to a market equilibrium based method, after exploring several different options [15]. 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 [9]. Similar to [9], 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 for another good cannot go down, hence the need to decrease the price of the second good does not arise.

As stated in [9], 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 [11], that does so. The algorithm of [9] 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 [9].

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 complementary slackness conditions for linear programs involve either primal or dual variables but not both. As a result, even though the dual growth process in [9] 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 [9] 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 [9] 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 applying 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 [9] 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 [9]. 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:

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

Our algorithm decreases this potential function by an inverse polynomial fraction in each phase, which can be implemented in strongly polynomial time. When  $\Phi$  drops all the way to zero, equilibrium is reached. However, since  $\Phi$  is a function of the initial money of the buyers, we only get a polynomial time algorithm. As in [9], the use of  $l_2$  norm in our algorithm makes its proof difficult. Very recently, [17] has shown that  $l_1$  norm-based potential functions do not suffice for establishing polynomial running time of the algorithm of [9], and hence justifies the use of the  $l_2$  norm in our algorithm.

Some of the notions introduced in [9] 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 [9]. However, at the referees' suggestion, we have made this paper completely self-contained, rather than referring the reader to [9] 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 there is 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  $j$  there is a function  $h_j^i : \mathbf{R}^+ \rightarrow \mathbf{R}^+$  which gives the utility that  $i$  derives as a function of the amount of good  $j$  that she receives. Her overall utility 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 and the example given in Section 3 should help with this.

Next, let us formally define arbitrary spending constraint utility functions in Fisher's model. For  $i \in B$  and  $j \in G$ , let  $f_j^i : [0, e(i)] \rightarrow \mathbf{R}^+$  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 [10]. 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  $\mathbf{p} = (p_1, \dots, p_n)$  for all the goods, consider baskets of goods that make  $i$  happiest (there could be many such baskets). We will say that  $\mathbf{p}$  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.,

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

- 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  $\mathbf{p} = (p_1, \dots, p_n)$ , we characterize optimal baskets for each buyer relative to  $\mathbf{p}$ . Define the *bang per buck* relative to prices  $\mathbf{p}$  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:  $Q_1, Q_2, \dots$ . For a class  $Q_l$ , define  $\text{value}(Q_l)$  to be the sum of the values of segments in it. At prices  $\mathbf{p}$ , 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  $\mathbf{p}$ ,  $i$ 's optimal allocation must contain goods corresponding to all segments in  $Q_1, \dots, Q_{k_i - 1}$ , 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  $\mathbf{p}$ ,  $Q_1, \dots, Q_{k_i - 1}$  are her *forced partitions*,  $Q_{k_i}$  is her *flexible partition*, and  $Q_{k_i + 1}, \dots$  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 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 the uniqueness of equilibrium prices, as claimed in Theorem 1. Suppose there are two equilibrium prices  $\mathbf{p}$  and  $\mathbf{p}'$  with  $\mathbf{p} \neq \mathbf{p}'$ , 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  $\mathbf{p}$  to  $\mathbf{p}'$ .

**Lemma 2** Consider an arbitrary buyer  $i$ . Let  $D$  and  $D'$  be (any) optimal bundles for  $i$  relative to prices  $\mathbf{p}$  and  $\mathbf{p}'$ . 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  $\mathbf{p}$ , in  $i$ 's forced and flexible allocations, respectively. Similarly, let  $Q'_1$  and  $Q'_2$  denote the set of segments, at prices  $\mathbf{p}'$ , in  $i$ 's forced and flexible allocations, respectively.

Since goods in  $S$  become more desirable under prices  $\mathbf{p}'$  as compared to prices  $\mathbf{p}$ , 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  $\bar{S}$  and if  $i$  prefers  $s'$  to  $s$  at prices  $\mathbf{p}$  then she must prefer  $s$  to  $s'$  at prices  $\mathbf{p}'$  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  $\mathbf{p}'$  as they do at prices  $\mathbf{p}$ . Since both these prices are equilibrium prices, the goods are fully sold at these prices, and hence the total money spent on the one available unit of each good must equal its price. Now, by the definition of  $\theta$ , goods in  $S$  have strictly less total value at prices  $\mathbf{p}'$  than at prices  $\mathbf{p}$ , 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  $\mathbf{p}$  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  $\text{value}(s)$  worth of good  $j$  from the total value of good  $j$  allocated to buyer  $i$ . In addition, at an intermediate point in the algorithm, w.r.t. current prices of goods, the buyer may be ambivalent between buying goods and saving some part of her unspent money. At this point, prices of goods cannot be raised further before an appropriate amount of money is set aside, to be finally returned to the buyer together with her optimal bundle of goods bought with the remaining money. Let  $\text{returned}(s), s \in \text{seg}(f_0^i)$ , denote the amount of money, corresponding to segment  $s$ , that will eventually be returned to  $i$ , where  $\text{returned}(s) \leq \text{value}(s)$ . If  $\text{returned}(s) > 0$ , 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  $\text{allocated}(j)$  denote the total amount of money spent on good  $j, j \neq 0$ , i.e., the total value of  $j$  already allocated and let  $\text{spent}(i)$  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,  $\text{value}(s)$  is added to  $\text{allocated}(j)$  and to  $\text{spent}(i)$ , and when  $\text{returned}(s)$  money is returned to  $i$ , corresponding to segment  $s \in \text{seg}(f_0^i)$ ,  $\text{returned}(s)$  is added to  $\text{spent}(i)$ . Also, define the *money left over* with buyer  $i, m(i) = e(i) - \text{spent}(i)$ .

Henceforth we will assume w.l.o.g. that for each buyer  $i$ , all segments under consideration have positive rate, i.e., segments of zero rate have been discarded. At any point in the run of the algorithm, the set of allocated segments for each buyer  $i$  must satisfy:

**Invariant 1:** At current prices  $\mathbf{p}$ , for each buyer  $i$ , her segments are partitioned into 3 sets,  $Q_i^-, Q_i^*, Q_i^+$ ; it is not essential for the sets to be non-empty. These 3 sets are called *allocated, current, and remaining partitions*, respectively. Relative to prices  $\mathbf{p}$ , all segments in  $Q_i^*$  have equal bang per buck, and they are strictly better than any segment in  $Q_i^-$ . The bang per buck of any segment in  $Q_i^+$  is at least as large as that of a segment in  $Q_i^*$ . The current set of fully allocated segments is precisely partition  $Q_i^+$ .

At termination of our algorithm, for each buyer  $i$ , if all segments in  $Q_i^+$  which have the same bang per buck as those in  $Q_i^*$  are moved into  $Q_i^*$ , then we will get the partition  $Q_{t_i}$ ; in case  $Q_i^* = \emptyset$  at termination, then the segments in  $Q_i^+$  having lowest bang per buck constitute  $Q_i^*$ .

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

## 5.1 The network $N$ , tight sets, and a characterization of equilibrium prices

We next define network  $N(\mathbf{p}, \mathbf{a}, \mathbf{s})$ , 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  $\text{good}(s) = j$ . The capacity of this edge,  $c_{ji}$ , equals  $\text{value}(s)$ . It also contains directed edges  $(s, j)$ , for each  $j \in G$  with capacity  $p_j - \text{allocated}(j)$ , and directed edges  $(i, t)$ , for each  $i \in B$  with capacity  $m(i)$ .

We have assumed that for each segment  $s, \text{value}(s)$  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  $\mathbf{a}$  is an integral vector.

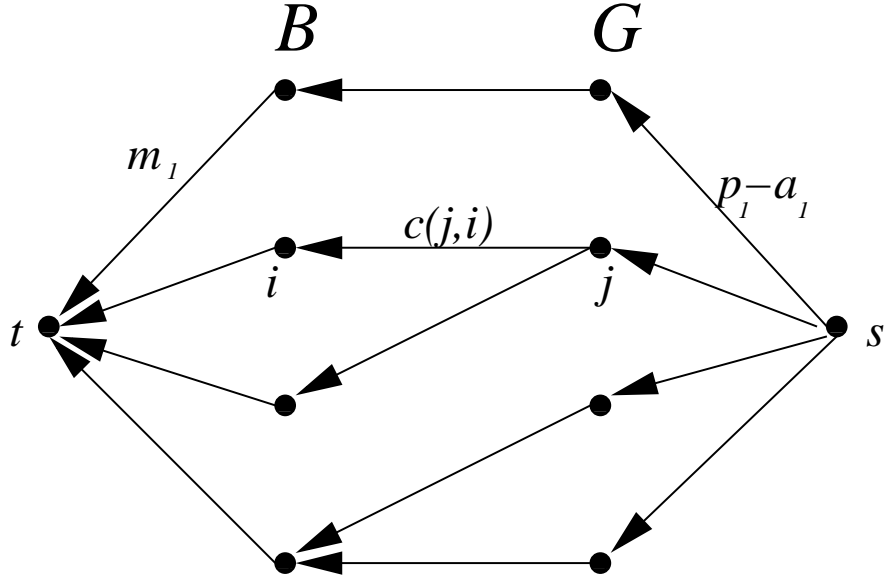


Figure 1: The network  $N(\mathbf{p}, \mathbf{a}, s)$ .

Throughout the algorithm, we will maintain the following:

**Invariant 2:**  $(s, G \cup B \cup t)$  is a min-cut<sup>1</sup> in network  $N(\mathbf{p}, \mathbf{a}, s)$ .

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.

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

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

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  $N(\mathbf{p}, \mathbf{a}, s)$ . For  $S \subseteq G$ , define

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

and define  $\text{bestT}(S)$  to be a maximal subset of  $\Gamma(S)$  that optimizes the above expression. Observe that  $\text{best}(S)$  is the capacity of the min-cut separating  $t$  from  $S$  in  $N(\mathbf{p}, \mathbf{a}, s)$ . Also observe that if  $T_1$  and  $T_2$  both 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  $\text{bestT}(S)$  is unique. We can now give a characterization of Invariant 2 in terms of cuts in the network.

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

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

<sup>1</sup>This section assumes familiarity with the theory of cuts and flows, e.g., see [1, 7].

**Proof :** If network  $N(\mathbf{p}, \mathbf{a}, \mathbf{s})$  satisfies Invariant 2, it supports a max-flow of value  $\mathbf{p}(G) - \mathbf{a}(G)$  that saturates all edges out of  $s$ . Since the flow going through set  $S \subseteq G$  is  $\mathbf{p}(S) - \mathbf{a}(S)$  and  $\text{best}(S)$  is the capacity of the min-cut separating  $t$  from  $S$  in  $N(\mathbf{p}, \mathbf{a}, \mathbf{s})$ , 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  $N(\mathbf{p}, \mathbf{a}, \mathbf{s})$ , with  $G_1, G_2 \subseteq G$  and  $B_1, B_2 \subseteq B$ ; it has size  $\mathbf{m}(B_1) + c(G_1, \Gamma(G_1) - B_1) + \mathbf{p}(G_2) - \mathbf{a}(G_2)$ . Let  $B'_1 = B_1 \cap \Gamma(G_1)$ . Then,

$$\text{best}(G_1) \leq \mathbf{m}(B'_1) + c(G_1, \Gamma(G_1) - B_1) \leq \mathbf{m}(B_1) + c(G_1, \Gamma(G_1) - B_1).$$

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

$$\begin{aligned} \mathbf{p}(G) - \mathbf{a}(G) &= [\mathbf{p}(G_1) - \mathbf{a}(G_1)] + [\mathbf{p}(G_2) - \mathbf{a}(G_2)] \\ &\leq \mathbf{m}(B_1) + c(G_1, \Gamma(G_1) - B_1) + \mathbf{p}(G_2) - \mathbf{a}(G_2), \end{aligned}$$

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

Given a feasible flow  $f$  in  $N$ ,  $R(f)$  will denote the residual graph w.r.t.  $f$ : For any two vertices  $u, v \in N$ , let the capacities of edges  $(u, v)$  and  $(v, u)$  in  $N$  be  $c \geq 0$  and  $c' \geq 0$ , respectively. Assume that  $f$  sends net flow  $f_e \geq 0$  from  $u$  to  $v$ . Then, in  $R(f)$ , edges  $(u, v)$  and  $(v, u)$  will have capacities  $c - f_e$  and  $c' + f_e$ , respectively. By a *residual edge or path* we mean an edge or path having positive capacity in  $R(f)$ .

A nonempty set  $S \subseteq G$  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  $S_1 \subseteq G$  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  $N(\mathbf{p}, \mathbf{a}, \mathbf{s})$  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,  $\mathbf{p}(S) - \mathbf{a}(S) = \min_{T \subseteq \Gamma(S)} \{\mathbf{m}(T) + c(S; \Gamma(S) - T)\}$ . Observe that if w.r.t. a max-flow in an *S-network* there is a residual path from  $S$  to  $t$  then there must also be a length 2 residual path. Using this fact, 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.  $\square$

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

Finally, we use the network  $N$  to give us a very useful way of characterizing equilibrium prices as well as a polynomial time algorithm for checking if given prices  $\mathbf{p}$  are equilibrium prices. This lemma provides the terminating condition for our algorithm.

To test if prices  $\mathbf{p}$  are equilibrium prices, first compute the forced, flexible and undesirable partitions of each buyer, as defined in Section 2. If the value of any good  $j$  allocated under forced

allocations exceeds its price  $p_j$ , then abort. Otherwise, construct the network  $N(\mathbf{p}, \mathbf{a}, \mathbf{s})$  with the flexible partitions providing the edges of  $N$ . The following lemma gives the desired characterization; its proof is obvious.

**Lemma 6** *Prices  $\mathbf{p}$  are equilibrium prices iff the both cuts  $(s, G \cup B \cup t)$  and  $(s \cup G \cup B, t)$  are min-cut in network  $N(\mathbf{p}, \mathbf{a}, \mathbf{s})$ .*

## 6 Balanced Flows

In this section, we will extend the notion of balanced flows from [9] to our setting, which is more involved because 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,  $N(\mathbf{p}, \mathbf{a}, \mathbf{s})$  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  $N$ . 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  $\boldsymbol{\gamma}(N, f) := (\gamma_1(N, f), \gamma_2(N, f), \dots, \gamma_{n'}(N, f))$ , where  $n' = |B|$ . Let  $\|v\|$  denote the  $l_2$  norm of vector  $v$ . A *balanced flow* in network  $N(\mathbf{p}, \mathbf{a}, \mathbf{s})$  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  $\boldsymbol{\gamma}_1$  and  $\boldsymbol{\gamma}_2$  are the surplus vectors w.r.t flows  $f_1$  and  $f_2$ , then  $(\boldsymbol{\gamma}_1 + \boldsymbol{\gamma}_2)/2$  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  $\|\cdot\|$  is strictly concave,  $\|\boldsymbol{\gamma}_1 + \boldsymbol{\gamma}_2\|/2$  is even smaller, leading to a contradiction.  $\square$

We will denote the unique surplus vector of a balanced flow in network  $N$  by  $\boldsymbol{\gamma}(N)$ . 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  $\boldsymbol{\gamma}_i(N, f) > \boldsymbol{\gamma}_j(N, f)$  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  $\boldsymbol{\gamma}_i(N, f) > 0$ , 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  $(j, t)$  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 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 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 in a way that results 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.  $\square$

## 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 edges 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  $N$ . By induction,  $f_1$  and  $f_2$  are max-flows in  $M_1$  and  $M_2$  respectively, with capacities reduced by flow  $h$ . Hence,  $f = h \cup f_1 \cup f_2$  is also a max-flow in  $N$ .  $\square$

**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,  $\alpha$ , 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  $\alpha$  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 at least

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

Now, since all  $L$  edges in  $E_L$  have the same surplus, each has surplus greater than  $\alpha$ . The lemma follows.  $\square$

**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 drops by at least one in each recursive call. 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  $N$ .  $\square$

Consider the union of flows  $h$  at all the levels of recursion. A well-chosen subset of these saturating flows will be routed as forced allocations in our algorithm.

## 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.

**Initialization:** Execute the following steps to find prices  $\mathbf{p}$  so that Invariants 1 and 2 hold with each buyer's current partition being her first partition w.r.t. these prices.

- 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  $\alpha_i$ , 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

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

It is easy to see that relative to prices  $\mathbf{p}$ , with no forced allocations,  $(s, G \cup B \cup t)$  is the only min-cut in  $N$ .

## 7.1 High level idea

Our algorithm starts with the (very low) prices  $\mathbf{p}$  found in the initialization and raises them iteratively, depleting the surplus money of buyers, until the condition of Lemma 6 holds, i.e., equilibrium is reached. The measure of progress of our algorithm is the  $l_2$  norm of the surplus vector w.r.t. a balanced flow in  $N$ . The algorithm is designed in such a way that this potential function decreases by an inverse polynomial fraction in polynomial time (more precisely, the time needed to execute a phase, which is defined below).

As detailed below, the algorithm raises prices only in certain iterations. If so, it raises the prices of a carefully chosen subset  $J \subseteq G$  of goods until one of the Invariants is threatened, i.e., fails to hold. It then makes suitable changes to the network  $N$  and set  $J$  so prices can be raised again. Within an iteration, it raises prices of goods in  $J$  in such a way that each buyer's current partition remains unchanged. We next describe how this is done.

Assume that buyer  $i$  has goods  $j$  and  $j'$  in her current partition, due to segments  $s \in \text{seg}(f_j^i)$  and  $s' \in \text{seg}(f_{j'}^i)$ . Then,

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

This suggests increasing the prices of goods in  $J$  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  $j \in J$  by  $x$ , initializing  $x = 1$ , and raising  $x$  continuously.

Recall the following definitions. Given a feasible flow  $f$  in  $N$ ,  $R(f)$  will denote the residual graph w.r.t.  $f$ : For any two vertices  $u, v \in N$ , let the capacities of edges  $(u, v)$  and  $(v, u)$  in  $N$  be  $c \geq 0$  and  $c' \geq 0$ . Assume that  $f$  sends net flow  $f_e \geq 0$  from  $u$  to  $v$ . Then in  $R(f)$ , edges  $(u, v)$  and  $(v, u)$  will have capacities  $c - f_e$  and  $c' + f_e$ , respectively. By a *residual edge or path* we mean an edge or path having positive capacity in  $R(f)$ .

A run of the algorithm is partitioned into *phases*, and each phase consists of two stages, *Stage I* and *Stage II*. Each stage is further partitioned into *iterations*. The prices of goods are raised in the iterations of Stage II but not Stage I. The algorithm keeps executing the stages alternately until one of the following two terminating conditions for a phase is met:

1. The left-over money of a buyer becomes zero.
2. The surplus money of a buyer in a balanced flow becomes zero (because the buyer is in  $\text{bestT}(S)$ , for a tight set  $S$ ).

At the start of a phase, the set  $I$  of buyers having maximum surplus w.r.t. a balanced flow is identified. Set  $J$  is initialized to contain all goods  $j$  having a residual edge to a buyer in  $I$ ; it is possible that  $J = \emptyset$ . During the iterations of Stage I, buyers may be moved from  $B - I$  to  $I$  and goods may be moved from  $G - J$  to  $J$ . Forced allocations are made of goods in  $G - J$  to buyers in  $I$ . This depletes the money of buyers in  $I$ . If as a result, some buyer in  $I$  has no left-over money, the entire phase terminates. The other possibility is that Stage I terminates and the algorithm moves to Stage II; this happens when each buyer  $i \in I$  has a residual edge from some good  $j \in J$ .

During each iteration of Stage II, prices of goods in  $J$  are raised using the mechanism described above. Also, during these iterations, buyers may be moved from  $B - I$  to  $I$ . If a set  $S \subseteq J$  goes tight and the set  $T = \text{bestT}(S) \neq \emptyset$ , the phase ends, since buyers in  $T$  will have surplus zero in a balanced flow in  $N$ . However, if  $T = \emptyset$ , the phase cannot be terminated. In this case, the goods of  $S$

are moved from  $J$  to  $G - J$ , since raising their prices will violate Invariant 2, and forced allocations are made corresponding to all (saturated edges) going from  $S$  to  $I$ . If as a result, some buyer  $i \in I$  has  $m(i) = 0$ , the phase terminates. If for each buyer  $i \in I$ ,  $Q_i^* \neq \emptyset$ , the algorithm proceeds with the next iteration within Stage II. Else, it goes to Stage I to update the current partitions of buyers in  $I$ .

Thus, throughout the phase, buyers are partitioned into two sets,  $I$  and  $B - I$ , and goods are goods are partitioned into two sets,  $J$  and  $G - J$ . Observe that whereas goods move from  $G - J$  to  $J$  and back, buyers can only move from  $B - I$  to  $I$ .

At any point when the prices of goods in  $J$  are being raised, for each buyer  $i \in I$ , her current partition satisfies  $Q_i^* \subseteq J$ , and for each  $i \in B - I$ , her current partition satisfies  $Q_i^* \subseteq G - J$ . As the prices of goods in  $J$  are raised and those of goods in  $G - J$  are held steady, the bang per buck of buyers in  $I$  decreases and those of buyers in  $B - I$  remains unchanged. As a result, goods in  $G - J$  are gradually becoming better for buyers in  $I$  and those in  $J$  are becoming worse for buyers in  $B - I$ . This causes the corresponding segments to migrate across partitions. The main job of the algorithm is to respond appropriately to these and other changes that happen dynamically to the partitions of each buyer, so that prices continue to be raised without violating the Invariants.

We give below an exhaustive list of all these changes from first principles. In the description of the algorithm, we indicate the steps it takes in response to each of these changes.

- **Change (a); update current partition:** For each buyer  $i \in B$ , if  $Q_i^* = \emptyset$  and  $m(i) > 0$ , then move the highest bang per buck segments from  $Q_i^-$  to  $Q_i^*$ .
- **Change (b); forced allocation:** For a buyer  $i \in I$ , a segment  $s \in Q_i^*$  is fully allocated and moved into  $Q_i^+$ . In this case,  $\text{good}(s)$  is currently in  $G - J$ .
- **Change (c); edge removal:** At the start of each iteration in Stage II, as soon as prices of goods in  $J$  are raised, for each buyer  $i \in B - I$ , goods in  $J$  become inferior to those in  $G - J$ . Each segment  $s \in Q_i^*$  with  $\text{good}(s) \in J$  is moved into  $Q_i^-$ .
- **Change (d); edge addition:** As prices of goods in  $J$  rise, at some point for a buyer  $i \in I$ , a segment  $s \in Q_i^-$  with  $\text{good}(s) \in (G - J)$  moves into  $Q_i^*$ . In this case we will say that segment  $s$  becomes *active-add*.
- **Change (e); deallocation:** As prices of goods in  $J$  rise, at some point for a buyer  $i \in B - I$ , a segment  $s \in Q_i^+$  with  $\text{good}(s) \in J$  moves into  $Q_i^*$ . In this case we will say that segment  $s$  becomes *active-deallocate*.

Let us specify the changes to be made to the network in allocating and deallocating segments. The operation of *allocating segment*  $s$  to buyer  $i$  consists of adding  $\text{value}(s)$  to  $\text{allocated}(j)$  and  $\text{spent}(i)$  and removing directed edge  $(j, i)$  from network  $N$ , where  $\text{good}(s) = j$ . The operation of *deallocating segment*  $s$  from buyer  $i$  consists of subtracting  $\text{value}(s)$  from  $\text{allocated}(j)$  and  $\text{spent}(i)$  and adding directed edge  $(j, i)$  to network  $N$ , where  $\text{good}(s) = j$ .

## 7.2 The algorithm for a phase

We first specify the following two subroutines:

### Initialize Phase

For each buyer  $i \in B$  such that  $Q_i^* = \emptyset$  and  $m(i) > 0$ , its current partition is updated and edges

added to  $N$ , i.e., Change (a) is executed. (By the conditions of Theorem 1, there are segments in  $Q_i^-$  to enable this.) Next, a balanced flow, say  $f$ , is computed in the current network,  $N(\mathbf{p}, \mathbf{a}, \mathbf{s})$ . If the cut  $(s \cup G \cup B, t)$  is saturated by  $f$ , then by Lemma 6 the current prices  $\mathbf{p}$  are equilibrium prices. If so, **halt** and output the current prices. Otherwise, let  $\delta$  be the maximum surplus of buyers w.r.t.  $f$ . Initialize  $I$  to be the set of buyers having surplus  $\delta$ . Initialize  $J$  to be the set of all goods having a residual edge to  $I$ .

### Update Sets

Find the set,  $T$ , of all buyers in  $B - I$  that have residual paths, in the sub-network  $N - \{s, t\}$ , to buyers in  $I$  and update:

$$I \leftarrow (I \cup T).$$

Next, find the set,  $S$ , of all goods in  $G - J$  that have residual edges to buyers in  $I$  and update:

$$J \leftarrow (J \cup S).$$

**Algorithm 1 (Algorithm for a Phase)**

1. Call subroutine **Initialize Phase**. (This subroutine initializes sets  $I$  and  $J$ .)

**Stage I**

2. (**New Iteration**) For each edge  $(j, i)$  such that  $j \in G - J$  and  $i \in I$ , execute Change (b), i.e., allocate this segment.
3. If  $\exists i \in I$  s.t.  $m(i) = 0$ , then **end the current phase**.
4. If  $\forall i \in I, Q_i^* \neq \emptyset$ , then go to Stage II.
5.  $\forall i \in I$  s.t.  $Q_i^* = \emptyset$ , update its current partition, i.e., execute Change (a).
6. Compute a balanced flow in  $N$  and call **Update Sets**.
7. Go to Step 2.

**Stage II**

8. (**New Iteration**) Remove all edges from  $J$  to  $B - I$ , i.e., execute Change (c).
9. Multiply the prices of goods in  $J$  by  $x$ , initialize  $x \leftarrow 1$ , and increase  $x$  continuously until:
  - a). **Event 1:** A set  $S, \emptyset \subset S \subseteq J$ , goes tight.
    - (i). If  $\text{bestT}(S) \neq \emptyset$ , then **end the current phase**.
    - (ii). If  $\text{bestT}(S) = \emptyset$ , then:

Move goods  $S$  from  $J$  to  $G - J$ .

For each edge  $(j, i)$  such that  $j \in S$  and  $i \in I$ , allocate this segment, i.e., execute Change (b).

If  $\exists i \in I$  s.t.  $m(i) = 0$ , then **end the current phase**.  
If  $\exists i \in I, Q_i^* = \emptyset$ , then go to Step 5 in Stage I.  
Else, go to Step 8.
  - b). **Event 2:** A segment becomes active-add or active-deallocate. If so:

Add an appropriate edge to  $N$  corresponding to each segment that becomes active-add, i.e., execute Change (d).

Deallocate corresponding to each segment that becomes active-deallocate, i.e., execute Change (e).Compute a balanced flow in  $N$  and call **Update Sets**.

For each edge  $(j, i)$  such that  $j \in G - J$  and  $i \in I$ , allocate this segment, i.e., execute Change (b).

Go to Step 8.

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 events takes place; the smaller of these is the event that happens first. For Event 2, 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 9.

Once the algorithm halts with equilibrium prices, equilibrium allocations are computed as follows. Compute a max-flow in  $N(\mathbf{p}, \mathbf{a}, \mathbf{s})$ . 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  $\text{value}(s)/p_j$ .

Observe that under the spending constraint step utility functions, a segment  $s \in \text{seg}(f_j^i)$  represents  $\text{value}(s)$  worth of good  $j$ ; the exact amount of good  $j$  it represents becomes clear only at the termination of the algorithm, when the equilibrium price of good  $j$  is determined. 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 to defining spending constraint utilities, which finesse these issues very effectively.

## 8 Proof of Correctness and Termination

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

**Lemma 12** *The algorithm maintains the Invariants throughout.*

**Proof :** The algorithm is designed to maintain the two Invariants – whenever one of them is about to become violated, it makes appropriate changes to  $N$  and  $J$  after which prices can again be increased without violating the Invariants.

In particular, allocating and deallocating segments do not affect Invariant 2 because these operations affect all  $s$ - $t$  cuts in  $N$  to the same extent. This is so because these operations subtract or add the same capacity, say  $c$ , on all three edges of an  $s$ - $t$  path.

Observe that the edges removed in Step 8 in Stage II do not carry any flow; indeed, if such an edge  $(j, i)$  carried non-zero flow then there would be a residual path from  $i$  to  $I$  and subroutine Update Sets would have moved  $i$  into  $I$ . Hence the removal of these edges does not violate Invariant 2.

Finally, a tight set  $S \subseteq J$  indicates that increasing prices any more will violate Invariant 2 and again, appropriate changes are made.  $\square$

The next lemma shows that each phase must make progress.

**Lemma 13** *In each phase, forced allocations are made or prices of goods are increased.*

**Proof :** The proof follows from the following three observations regarding the first time Stage I and Stage II are executed in the phase:

- The first iteration of Stage I cannot terminate in Step 3, since all buyers  $i \in I$  have surplus  $\delta$  at this moment (independent of whether forced allocations were made in Step 2).

- Each iteration in Stage I, other than the ultimate one, makes forced allocations.
- At the start of the first iteration in Stage II, there are no active-add or active-deallocate segments. Further, there are no tight sets because each buyer  $i \in I$  has  $m(i) \neq 0$  and has a residual edge from a good in  $J$ , i.e.,  $Q_i^* \neq \emptyset$ . Hence, there are no obstacles to increasing prices.

Now, if no allocations are made in Stage I (which must have only one iteration), then the algorithm must execute Stage II and increase prices of goods. The lemma follows.  $\square$

**Lemma 14** *If a phase terminates with tight set  $S \subseteq J$  with  $\text{bestT}(S) \neq \emptyset$ , then the prices of goods in  $S$  are rational numbers with denominators  $\leq \Delta$ .*

**Proof :** By definition of tight set,

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

Consider the subgraph induced on the bipartition  $(S, T)$  by the current edges, after making them undirected. 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  $\mathbf{p}(S) = p_j c/d$  where  $c \leq \Delta$ . On the other hand,  $\mathbf{p}(S) = \mathbf{m}(T) + \mathbf{a}(S) - c[S; \Gamma(S) - T]$ . Since each term on the right hand side is integral, so is  $\mathbf{p}(S)$ . Therefore,

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

hence proving the lemma.  $\square$

On the other hand, if Stage II terminates with tight set  $S \subseteq J$  with  $\text{bestT}(S) = \emptyset$ , then the prices of goods in  $S$  are integral. Of course, if a phase never executes Stage II, prices of goods remain unchanged.

**Corollary 15** *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 14,  $q \leq \Delta$  and  $s \leq \Delta$ . Therefore the increase in price of  $j$ ,

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

$\square$

**Lemma 16** *The total number of iterations in a phase is bounded by  $3z + 2$ .*

**Proof :** Each iteration, other than the very first and last, involve allocations, deallocations, edge additions or removals. We will show that each segment can participate in at most three of these operations, hence establishing the bound.

We start by showing how a segment can participate in three operations. Let  $s$  be a segment for buyer  $i$  and good  $j$ , and assume that  $i \in (B - I)$  and  $j \in J$  and that edge  $(j, i)$  is initially in the network. First, edge  $(j, i)$  is removed or deallocated (due to Change (c) or (e)). Second,  $i$  moves to  $I$ . Third,  $j$  moves to  $G - J$ ; this happens because  $j$  is in a tight set  $S$  and  $\text{bestT}(S) = \emptyset$ . Fourth, edge  $(j, i)$  is added (due to Change (a) or (d)). Fifth,  $j$  moves back to  $J$ . Finally, segment  $s$  is allocated because once again,  $j$  is in a tight set  $S$  and  $\text{bestT}(S) = \emptyset$ .

Next, observe that since  $i$  cannot move back to  $(B - I)$ ,  $s$  will not be deallocated in the present phase. Hence a fourth operation on this segment is not possible.  $\square$

**Theorem 17** *The algorithm terminates with an equilibrium.*

**Proof :** By Lemma 12, the algorithm maintains Invariants 1 and 2 throughout, and by Lemma 13, each phase must make progress, i.e., either decreasing the left-over money of buyers due to allocations or increasing the prices of goods. At first sight it appears that a deallocation will undo the first type of progress. However, observe that the balanced flow executed right after a deallocate step has the option of sending the flow right back (however, if rerouting an equivalent amount of flow results in a further decrease in the  $l_2$ -norm of the surplus vector, the algorithm will of course execute that option). Hence, the surplus of buyers must eventually vanish, i.e.,  $(s \cup G \cup B, t)$  becomes a min-cut in network  $N(\mathbf{p}, \mathbf{a}, \mathbf{s})$ . Finally, by Lemma 6 the prices at this point must be equilibrium prices.  $\square$

Since the algorithm never decreases the price of any good, we get:

**Corollary 18** *Let  $\mathbf{p}$  be any prices satisfying the Invariants and let  $\mathbf{q}$  be the unique equilibrium prices. Then, for each good  $j \in G$ ,  $p_j \leq q_j$ .*

## 9 Computing $x^*$ via Min-Cuts in Parametric Networks

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

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

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

*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  $W'(\mathbf{p}, \mathbf{a}, \mathbf{s})$ . Let the min-cut maximizing the  $s$  side be  $(s \cup J_1 \cup I_1, J_2 \cup I_2 \cup t)$ . Since  $W'(\mathbf{p}, \mathbf{a}, \mathbf{s})$  satisfies Invariant 2 at  $x = \beta$ , for any set  $S \subseteq J$ ,

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

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

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

The lemma follows. □

**Lemma 20** *The following hold:*

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

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

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  $W'(\mathbf{p}, \mathbf{a}, \mathbf{s})$ . First observe that  $S^* \subseteq J_2$  contradicts the minimality of this cut: since  $\mathbf{p}(S^*) \cdot x - \mathbf{a}(S^*) > \text{best}(S^*)$ , 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  $\text{best}(S_1) + \text{best}(S_2) \leq \text{best}(S^*)^2$ . To achieve a similar effect even if  $\Gamma(S_1) \cap \Gamma(S_2) \neq \emptyset$  let us define for  $S \subseteq J_2$ :

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

and let us define  $\text{bestT}'(S)$  to be a maximal subset of  $\Gamma(S)$  optimizing the above expression. (In fact  $\text{bestT}'(S)$  is unique; this follows by the same argument given for showing that  $\text{bestT}(S)$  is unique.) Now observe that

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

Hence,

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

If  $\text{best}'(S_2) < x \cdot \mathbf{p}(S_2) - \mathbf{a}(S_2)$ , then a smaller cut can be found by moving  $S_2$  into  $J_1$ , and moving  $\text{bestT}'(S_2)$  from  $I_2$  to  $I_1$ . Therefore,

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

---

<sup>2</sup>Actually, this inequality holds with equality; we have relaxed it to an inequality in order to lead up to the next fact.

Combining with the previous inequality, we get

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

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

For  $i \in I$ , denote the sum of capacities of edges incident at  $i$  in  $W(\mathbf{p}, \mathbf{a}, \mathbf{s})$  by  $c(i)$ . Define  $m'(i) = \min\{m(i), c(i)\}$ , and  $\mathbf{m}'$  to be the vector consisting of  $m'(i), i \in I$ . Observe that replacing  $\mathbf{m}$  by  $\mathbf{m}'$  in  $W(\mathbf{p}, \mathbf{a}, \mathbf{s})$  does not change the min-cut or its capacity. Define  $W''(\mathbf{p}, \mathbf{a}, \mathbf{s})$  to be the network obtained by replacing  $\mathbf{m}$  by  $\mathbf{m}'$  in  $W(\mathbf{p}, \mathbf{a}, \mathbf{s})$ . The reason for working with  $\mathbf{m}'$  is that in  $W''(\mathbf{p}, \mathbf{a}, \mathbf{s})$  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  $x = (\mathbf{m}'(I) + \mathbf{a}(J))/\mathbf{p}(J)$  and let the minimal min-cut in  $W''(\mathbf{p}, \mathbf{a}, \mathbf{s})$  (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  $x$ ,  $x \cdot \mathbf{p}(J) - \mathbf{a}(J) = \mathbf{m}'(I)$  and by the property of  $\mathbf{m}'$  stated above,  $\text{best}(J) = \mathbf{m}'(I)$ . Therefore  $\text{best}(J) = x^* \cdot \mathbf{p}(J) - \mathbf{a}(J)$ , 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  $\mathbf{m}'(I)$ . For the chosen value of  $x$ , the capacity of  $(s, J \cup I \cup t)$  is  $x \cdot \mathbf{p}(J) - \mathbf{a}(J) = \mathbf{m}'(I)$  contradicting the fact that it is not a min-cut. Hence  $J_1$  is a proper subset of  $J$ .  $\square$

In the setting of Lemma 21, suppose  $x > x^*$  and  $(s \cup J_1 \cup I_1, J_2 \cup I_2 \cup t)$  is the min-cut in  $W''(\mathbf{p}, \mathbf{a}, \mathbf{s})$ . From the network  $W(\mathbf{p}, \mathbf{a}, \mathbf{s})$  we will construct a new network  $\overline{W}(\mathbf{p}, \mathbf{a}, \mathbf{s})$  which will reveal  $x^*$  and  $S^*$ . The vertices of the new network are  $\{s\} \cup I'_1 \cup J_1 \cup \{t\}$ , where  $I'_1 = I_1 \cup \{v\}$ , where  $v$  is a new vertex (which, in a sense, is meant to replace all the buyers in  $I_2$ ). Corresponding to each edge  $(j, i)$ ,  $j \in J_1$ ,  $i \in I_2$  of  $W(\mathbf{p}, \mathbf{a}, \mathbf{s})$  this network has the edge  $(j, v)$  and the capacity of this edge is the same as that of  $(j, i)$ . In addition, it has the edge  $(v, t)$  with capacity equal to the sum of money of all buyers that  $v$  replaces, i.e., all the buyers in  $I_2$ . All the remaining edges of  $W(\mathbf{p}, \mathbf{a}, \mathbf{s})$  from  $s$  to  $J_1$ ,  $J_1$  to  $I_1$ , and  $I_1$  to  $t$  appear in the new network with the original capacities.

For  $S \subseteq J_1$ ,  $\text{best}(S)$  and  $\text{bestT}(S)$  will refer to these entities in the network  $W(\mathbf{p}, \mathbf{a}, \mathbf{s})$ . For the purpose of the next lemma, for each  $S \subseteq J_1$ , define

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

w.r.t. network  $\overline{W}(\mathbf{p}, \mathbf{a}, \mathbf{s})$ . Also define  $\text{bestT}'(S)$  to be the (unique) maximal subset of  $\Gamma(S)$  that optimizes the above expression.

**Lemma 22** 
$$x^* = \min_{\emptyset \neq S \subseteq J_1} \frac{\text{best}'(S) + \mathbf{a}(S)}{\mathbf{p}(S)},$$
 and the unique maximal set minimizing the above expression is  $S^*$ .

**Proof :** By Lemma 20,  $S^* \subseteq J_1$ , and therefore,  $\text{best}'(S^*)$  is well defined in the network  $\overline{W}(\mathbf{p}, \mathbf{a}, \mathbf{s})$ . We will show that for  $S \subseteq J_1$ ,  $\text{best}'(S) \geq \text{best}(S)$  and that  $\text{best}'(S^*) = \text{best}(S^*)$ . The present lemma will then follow from Lemma 19.

Since edge  $(v, t)$  has a very large capacity,  $v$  will not be in  $\text{bestT}'(S)$  for any  $S \subseteq J_1$ . Furthermore, every choice of  $T \subseteq (\Gamma(S) - \{v\})$  that is available in network  $\overline{W}(\mathbf{p}, \mathbf{a}, \mathbf{s})$  is also available in the network  $W(\mathbf{p}, \mathbf{a}, \mathbf{s})$  and the expression  $\mathbf{m}(T) + c(S; \Gamma(S) - T)$  has the same value in both cases. Now, since there are more choices for  $T$  in network  $W(\mathbf{p}, \mathbf{a}, \mathbf{s})$  than in network  $\overline{W}(\mathbf{p}, \mathbf{a}, \mathbf{s})$ , we get that for  $S \subseteq J_1$ ,  $\text{best}'(S) \geq \text{best}(S)$ .

Next, we show that  $\text{best}'(S^*) = \text{best}(S^*)$ . Suppose that  $\text{bestT}(S^*) = T \subseteq I_1$ . If  $T \subseteq I_1$ , the statement is obvious. So, assume that  $T \not\subseteq I_1$ . By definition of  $\text{best}(S)$  and  $\text{bestT}(S)$ ,  $\mathbf{m}(T - I_1) \leq c(S^*; (\Gamma(S^*) \cap (T - I_1)))$ . Next, let us argue that this inequality must hold with equality. If the inequality were strict, then moving  $(T - I_1)$  into  $I_1$  would decrease the capacity of the cut  $(s \cup J_1 \cup I_1, J_2 \cup I_2 \cup t)$  in network  $W''(\mathbf{p}, \mathbf{a}, \mathbf{s})$ , thereby contradicting the fact that we started with a min-cut. But if we have equality,  $\text{best}'(S^*) = \text{best}(S^*)$ .

The lemma follows.  $\square$

**Theorem 23**  $x^*$  and  $S^*$  can be found using at most  $n$  max-flow computations.

**Proof :** Let  $x = (\mathbf{m}'(I) + \mathbf{a}(J))/\mathbf{p}(J)$  and compute a min-cut in  $W''(\mathbf{p}, \mathbf{a}, \mathbf{s})$ . If  $(s, J \cup I \cup t)$  is a min-cut in  $N''(\mathbf{p}, \mathbf{a}, \mathbf{s})$ , then by Lemma 21  $x^* = x$  and  $S^* = J$ . Otherwise,  $x > x^*$ . Now, by Lemma 22, it is sufficient to recurse on the network  $\overline{W}(\mathbf{p}, \mathbf{a}, \mathbf{s})$ , which has fewer goods.  $\square$

## 10 Establishing Polynomial Running Time

In this section, we will establish polynomial running time for our algorithm by showing that the  $l_2$ -norm of the surplus vector drops by an inverse polynomial fraction in each phase.

In the next lemma we will assume that  $\mathbf{p}$  denotes prices at the end of the iteration and  $N(\mathbf{p})$  the network obtained after making all changes to edges during the iteration. Flow  $f$  is meant to be the flow at the beginning of the iteration and  $f^*$  the flow at the end of the iteration after making edge changes and finding a balanced flow.

**Lemma 24** If  $f$  and  $f^*$  are respectively a feasible and a balanced flow in  $N(\mathbf{p})$  such that  $\gamma_i(\mathbf{p}, f^*) = \gamma_i(\mathbf{p}, f) - \delta$ , for some  $i \in B$  and  $\delta > 0$ , then  $\|\gamma(\mathbf{p}, f^*)\|^2 \leq \|\gamma(\mathbf{p}, f)\|^2 - \delta^2$ .

**Proof :** Suppose we start with  $f$  and get a new flow  $f'$  by decreasing the surplus of  $i$  by  $\delta$ , 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  $\delta^2$  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  $i_1, i_2, \dots, i_k$ , we have  $\gamma_{i_l}(f') = \gamma_{i_l}(f) + \delta_l$ , s.t.  $\delta_1, \delta_2, \dots, \delta_k > 0$  and  $\sum_{l=1}^k \delta_l \leq \delta$ . Moreover, for all  $l$ , there is a path from  $i$  to  $i_l$  in  $R(\mathbf{p}, f^*)$ . 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 25,  $\|\gamma(\mathbf{p}, f')\|^2 \leq \|\gamma(\mathbf{p}, f)\|^2 - \delta^2$  and since  $f^*$  is a balanced flow in  $N(\mathbf{p})$ ,  $\|\gamma(\mathbf{p}, f^*)\|^2 \leq \|\gamma(\mathbf{p}, f')\|^2$ .  $\square$

**Lemma 25** If  $a \geq b_i \geq 0, i = 1, 2, \dots, n$  and  $\delta \geq \sum_{j=1}^n \delta_j$  where  $\delta, \delta_j \geq 0, j = 1, 2, \dots, n$ , then  $\|(a, b_1, b_2, \dots, b_n)\|^2 \leq \|(a + \delta, b_1 - \delta_1, b_2 - \delta_2, \dots, b_n - \delta_n)\|^2 - \delta^2$ .

**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 a total of  $k$  iterations in all its stages, and that  $N_t$  denotes the network at the end of iteration  $t$ . Let  $f_t$  be a balanced flow in  $N_t$  and let  $I_t$  denote the set  $I$  in the network  $N_t$ , for  $0 \leq t \leq k$ .

**Lemma 26**  $f_t$  is a feasible flow in  $N_{t+1}$ , for  $0 \leq t < k$ .

**Proof :** 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.
- Deallocating a segment. To justify this, observe that the balanced flow executed right after deallocating a segment  $s$  will simply undo the deallocation, in case it does not have a way of rerouting flow to decrease the  $l_2$ -norm of the surplus vector.

The lemma follows. □

Lemmas 24 and 26 yield:

**Corollary 27**  $\|\gamma(N_t)\|$  is monotonically decreasing with  $t$ .

Let  $\delta_t$  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 28** If  $\delta_{t-1} > \delta_t$  then there exists an  $i \in I_{t-1}$  such that  $\gamma_i(\mathbf{p}_{t-1}) - \gamma_i(\mathbf{p}_t) \geq \delta_{t-1} - \delta_t$ .

**Proof :** Consider the residual network  $R(\mathbf{p}_t, f)$  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  $R(\mathbf{p}_t, f)$  and therefore, by Property 1,  $\gamma_v(\mathbf{p}_t) \geq \gamma_i(\mathbf{p}_t)$ . This means that the minimum surplus  $\delta_t$  is achieved by a vertex  $i$  in  $I_{t-1}$ . Hence the surplus of vertex  $i$  decreases by at least  $\delta_{t-1} - \delta_t$  during iteration  $t$ . □

**Lemma 29** If  $\delta_t > \delta_{t+1}$  then  $\|\gamma(N_t)\|^2 - \|\gamma(N_{t+1})\|^2 \geq (\delta_t - \delta_{t+1})^2$ , for  $0 \leq t < k$ .

**Proof :** By Lemma 28, 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 26,  $f_t$  is a feasible flow in  $N_{t+1}$ . Finally, Lemma 24 gives the desired conclusion. □

**Lemma 30**  $\|\gamma(N_0)\|^2 - \|\gamma(N_k)\|^2 \geq \frac{\delta^2}{3z + 2}$ .

**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 27, each of these terms is nonnegative.

Consider only those terms in which the difference  $\delta_t - \delta_{t+1} > 0$ . The sum of their squares is minimized when all these differences are equal. Using Lemma 29 and the fact that  $\delta_0 = \delta$  and  $\delta_k = 0$ , yields

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

By Lemma 16,  $k \leq 3z + 2$ . The lemma follows.  $\square$

**Lemma 31** *In a phase, the  $l_2^2$ -norm of the surplus vector drops by a factor of*

$$\left(1 - \frac{1}{n(3z + 2)}\right).$$

**Proof :** From Lemma 30 and the fact that  $\|\gamma(N_0)\|^2 \leq n\delta^2$ ,

$$\begin{aligned} \|\gamma(N_k)\|^2 &\leq \|\gamma(N_0)\|^2 - \frac{n\delta^2}{n(3z + 2)} \leq \|\gamma(N_0)\|^2 - \frac{\|\gamma(N_0)\|}{n(3z + 2)} \\ &\leq \|\gamma(N_0)\|^2 \left(1 - \frac{1}{n(3z + 2)}\right). \end{aligned}$$

The lemma follows.  $\square$

**Theorem 32** *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.*

**Proof :** By Lemma 31, the square of the surplus vector drops by a factor of two 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 15, 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 16 each phase consists of at most  $3z + 2$  iterations and by Theorem 23 each iteration requires  $n$  max-flow computations. The theorem follows.  $\square$

## 11 Buyers with Utility for Money

Finally, we allow buyers to have utility for money, as 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 network  $N(\mathbf{p}, \mathbf{a}, \mathbf{s})$ .

As prices of goods are raised, the current bang per buck,  $\alpha_i$ , of each buyer  $i \in I$  keeps decreasing. If for some buyer  $i \in I$ ,  $\alpha_i$  decreases to the point where she is equally happy leaving with money corresponding to segment  $s \in \text{seg}(f_0^i)$ , 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 set  $I$  in a subsequent phase, the algorithm will first attempt to return the rest of  $\text{value}(s)$  to  $i$ .

The following event is executed as the first event in Step 9 of the algorithm for a phase.

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

Raise  $\text{returned}(s)$  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  $i \in \text{bestT}(S)$ ,  $\text{best}(S)$  is decreasing as  $\text{returned}(s)$  is raised.)  
If so, terminate the current phase and start the next phase.
- **Event 0(b):**  $\text{returned}(s) = \text{value}(s)$  (i.e., the money corresponding to segment  $s$  has been fully returned), or  $e(i) - \text{spent}(i) = 0$  (i.e., buyer  $i$  has no left-over money).  
Go to Step 10 and continue raising the prices of goods in  $J$ .

Let  $y^*$  denote the value of  $\text{returned}(s)$  at which Event 0(a) occurs. Next, we give a procedure for determining which of the two events occurs, and if Event 0(a) happens, we will give a method for computing  $y^*$ .

Let  $v'(s)$  denote  $(\text{value}(s) - \text{returned}(s))$  just before Event 0 was executed, i.e., the unreturned part of this segment, and  $r'$  denote the minimum of  $(e(i) - \text{spent}(i))$  (i.e., the unspent money of  $i$ ) and  $v'(s)$ . Compute  $\mathbf{p}'$ , the prices of all goods at the moment Event 0 occurs. Let  $\mathbf{a}$  denote all forced allocations made so far. Let  $\mathbf{s}$  denote the vector of money spent just before Event 0 was executed.

Obtain  $\mathbf{s}'$  from  $\mathbf{s}$  by adding  $r'$  to  $\text{spent}(i)$ , i.e., assume that  $r'$  money is returned to  $i$ , corresponding to segment  $s$ . Construct network  $W(\mathbf{p}', \mathbf{a}, \mathbf{s}')$  on vertices  $\{s\} \cup J \cup I \cup \{t\}$  and check if  $(s, J \cup I \cup t)$  is a min-cut in it. If so, Event 0(b) occurs, i.e.,  $r'$  money corresponding to segment  $s$  can be returned to  $i$ . It may be that simultaneously a set goes tight also; if so, the algorithm will realize this as soon as it tries to raise  $x$  further. If  $(s, J \cup I \cup t)$  is not a min-cut in the network, Event 0(a) occurs. If so, let  $(s \cup J_1 \cup I_1, J_2 \cup I_2 \cup t)$  be a maximal min-cut in the network. The next lemma shows how to obtain  $y^*$  in this event.

**Lemma 33** *If Event 0(a) occurs,*

$$y^* = \mathbf{m}(I_1) + c(J_1; \Gamma(J_1) - I_1) - (\mathbf{p}'(J_1) - \mathbf{a}(J_1)),$$

where  $\mathbf{m}(I_1)$  is computed by using the money spent according to vector  $\mathbf{s}$ .

**Proof :** Let  $W_1$  denote the network  $W(\mathbf{p}', \mathbf{a}, \mathbf{s}')$  defined above. Define a second network  $W_2$  as follows. Obtain  $\mathbf{s}''$  from  $\mathbf{s}$  by adding  $y^*$  to  $\text{spent}(i)$ , i.e., assume that  $y^*$  money is returned to  $i$ , corresponding to segment  $s$ . Let  $W_2$  denote the corresponding network, i.e.,  $W(\mathbf{p}', \mathbf{a}, \mathbf{s}'')$ . Clearly,  $W_2$  will have a tight set, say  $S$ , and  $i \in \text{bestT}(S)$ . Let  $C$  be a maximal min-cut in  $W_2$ ; since  $i \in \text{bestT}(S)$ ,  $i$  will be on the  $s$ -side of this cut.

The only difference between  $W_1$  and  $W_2$  is that the left-over money of  $i$  in  $W_1$  is smaller than that in  $W_2$  by  $r' - y^*$ . Therefore, it is easy to see that cut  $C$  will be a maximal min-cut in  $W_1$  as well, i.e.,  $C$  is the same as the cut  $(s \cup J_1 \cup I_1, J_2 \cup I_2 \cup t)$ . Moreover, the capacity of this cut is smaller in  $W_1$  by precisely  $r' - y^*$ .

Since network  $W_2$  satisfies Invariant 2, the capacity of cut  $C$  in  $W_2$  is the same as that of cut  $(s, J \cup I \cup t)$ , which is  $\mathbf{p}'(G) - \mathbf{a}(G)$ . The capacity of cut  $C$  in  $W_1$  is

$$\mathbf{m}(I_1) - r' + c(J_1; \Gamma(J_1) - I_1) + \mathbf{p}'(J_2) - \mathbf{a}(J_2).$$

Finally, using the fact that the difference of these capacities is  $r' - y^*$ , we get the expression for  $y^*$  given in the statement of this lemma.  $\square$

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

**Lemma 34** *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 subgraph induced on  $(S, T)$  by the current edges, after making them undirected, 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,  $\tau$ , in  $(S, T)$ .

Observe that when  $\text{rate}(s)$  became equal to  $\alpha_i$ , 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  $\text{rate}(i, j)$  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  $\tau$ , 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  $\Delta$ .  $\square$

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  $\alpha_i$  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 32 holds for the enhanced model as well, as long as  $z$  is taken to be the total number of segments having positive rate in all utility functions specified in the input, including those for money.

## 12 Discussion

The remarkable convex program given by Eisenberg and Gale [11] 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.

**Note added March 2010:** Recently, Devanur [8] has found a convex program for the linear Fisher markets that is very different from the Eisenberg-Gale program. A natural generalization of this program yields a convex program for Fisher markets with spending constraint step utility functions, thereby settling the conjecture stated above. As a result, this paper is also fits into a research theme proposed in recent years by the author, i.e., finding efficient combinatorial algorithms for solving special classes of nonlinear convex programs, e.g., see [18].

## 13 Acknowledgments

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, and Aranyak Mehta for superbly playing the role of critic and sounding board when I was working out the toughest parts of the proof. My most profound gratitude goes to the superb referees of *MOR* who provided extremely detailed and insightful comments, over several iterations, to previous versions of this paper and helped bring it to its current state.

## 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] X. Chen, D. Dai, Y. Du, and S.-H. Teng. Settling the complexity of arrow-debreu equilibria in markets with additively separable utilities. In *FOCS*, 2009.
- [6] X. Chen and S.-H. Teng. Spending is not easier than trading: on the computational equivalence of Fisher and Arrow-Debreu equilibria. *Journal of the ACM*, 56(3), 2009.
- [7] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. John Wiley and Sons, Inc., N.Y., 1998.
- [8] N. Devanur. Personal communication, 2010.
- [9] 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: <http://www-static.cc.gatech.edu/~vazirani/market.ps>.
- [10] N. Devanur and V. V. Vazirani. The spending constraint model for market equilibrium: Algorithmic, existence and uniqueness results. In *Proceedings of 36th STOC*, 2004.
- [11] E. Eisenberg and D. Gale. Consensus of subjective probabilities: the Pari-Mutuel method. *The Annals of Mathematical Statistics*, 30:165–168, 1959.
- [12] D. Gale. Piecewise linear exchange equilibrium. *Journal of Mathematical Economics*, 4:81–86, 1977.
- [13] 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.
- [14] N. Megiddo and C.H. Papadimitriou. On total functions, existence theorems, and computational complexity. *Theoretical Computer Science*, 81:317–324, 1991.
- [15] N. Nisan, J. Bayer, D. Chandra, T. Franji, R. Gardner, Y. Matias, N. Rhodes, M. Seltzer, D. Tom, and H. Varian. Google’s auction for TV ads. In *ICALP*, 2009.
- [16] D. Patinkin. *Money, Interest, and Prices. An Integration of Monetary and Value Theory*. Harper and Row, N.Y., 1965.
- [17] V. V. Vazirani. Nash bargaining via flexible budget markets. Submitted to *JACM*, 2008.
- [18] V. V. Vazirani. Seeking combinatorial algorithms for convex programs. In *Noam Nisan’s Algorithmic Game Theory Blog*, January 12, 2010.

- [19] V. V. Vazirani and L. Wang. Continuity properties of equilibria in some Fisher and Arrow-Debreu market models. In *Proceedings of The 5th Workshop on Internet and Network Economics*, 2009.
- [20] V. V. Vazirani and M. Yannakakis. Market equilibria under separable, piecewise-linear, concave utilities. In *Proceedings of The First Symposium on Innovations in Computer Science*, 2010.