UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

An Algorithm to Find a Core Point for a Two-Sided Matching Model

Thomas Quint

March 1988

CAM Report 88-03

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90024-1555

AN ALGORITHM TO FIND A CORE POINT FOR A TWO-SIDED MATCHING MODEL

by

Thomas Quint

Abstract

In this paper, we present an algorithm which finds a core point for the two-sided matching model of Demange-Gale (1985). The algorithm is similar to that of Crawford-Knoer (1981), but runs faster.

1. Introduction

A two sided matching market (TSMM) is a game in which there are two types of agents, and the essential coalitions are singletons and doubletons containing one agent of each type. Over the years, they have become an important part of economic theory. One reason for this is that they have been deemed worthy models of economic markets with indivisible goods. As far back as 1962, Gale and Shapley modeled marriage as a matching market. A decade later, Shapley and Shubik (1972) adopted a similar model for their housing market. And still later, Crawford and Knoer (1981) defined labor markets in these terms.

In all these instances, the relevant solution concept is that of the core. Simply put, the core is the set of economic allocations where no coalition of agents can improve their lot on their own. Herein lies another reason for the study of these games; the fact that their cores have many "nice" properties. For instance, their cores are always nonempty (Kaneko 1982, Quinzii 1984). Indeed, much of the literature (Gale & Shapley, Shapley & Shubik, Crawford & Knoer, Kelso & Crawford 1982) relates relatively simple algorithms which calculate core points.

This paper too is about an algorithm to find a core point.

The model is that of Demange and Gale (1985). It is "general", in that it is not assumed (as in Shapley-Shubik) that "utility is identified with money". Thus, when

a pair $(i\ j)$ form a coalition, instead of dividing up a total *utility*, they split a total amount of *money*. The valuation that an agent has for money is not constant—instead, it can depend on who the agent is, who he is matched with, or how much money he already has.

Even in this more general setting, much is known about the core. First, we repeat that it is nonempty. Also, the projection of the core onto the space of welfares of players on one side of the market is a sublattice. This implies the existance of a "seller optimum" $\underline{u}^H$ and a "buyer optimum" $\underline{u}^L$. These points are significant not only in that they represent the "extremal" outcomes of the economy, but also because Demange and Gale have shown that in a certain sense they are "nonmanipulable". However, it is not true that the allocations $\underline{u}^H$ and $\underline{u}^L$ are feasible under *every* core assignment. Thus, if we wish to calculate $\underline{u}^H$, say, it is a nontrivial problem to find a core assignment μ under which $\underline{u}^H$ is feasible.

The logic of the paper is as follows. First, we represent the core as the set of solutions to a system of mathematical equations M. We then define a relaxation R of this system which is easy to solve. Under mild assumptions, we define a procedure which moves from solution to solution of R, eventually stopping at one for M.

The algorithm turns out to be very similar to Crawford and Knoer's (1981), but we argue that ours is faster. We prove that if μ^* is the assignment output by the algorithm, then $(\mu^*, \underline{u}^H)$ is in the core. This simplifies the problem of finding $\underline{u}^H$. Finally, for the Shapley-Shubik market, which is a special case of this model, we prove that the "mild assumptions" above are met if the "associated assignment linear program" (Section 4) is nondegenerate.

The paper is organized as follows. Section 2 describes the model and gives background results. In section 3, we describe Crawford and Knoer's algorithm, and comment on a major drawback. Finally, in section 4, we describe our algorithm, prove some properties concerning its convergence, and compare it to Crawford and

Knoer's.

2. The Model[1]

The Demange-Gale setup is a model of a market for large, indivisible goods. Each of the agents only considers owning zero or one such object, and different agents may value the same object differently.

To facilitate trading among the agents, there is a second, completely divisible good called money. Money can only be exchanged between agents who trade with each other. This is an important assumption because, if free transferral of money among *all* agents is allowed, the core is in general empty.

Now let us describe the model in terms of a labor market, where the "large indivisible goods" are jobs. The market contains n employers (firms) and n employees (workers).[2]. Let I and J denote these sets. Each employer has one job opening, and, likewise, employees only consider taking one job or remaining unemployed. The ith employer places a value of $u_{ij}(-x)$ on the prospect of hiring the jth employee at a salary of x, where $u_{ij} : \Re \longrightarrow \Re$ is a strictly increasing, onto function. Similarly, $v_{ij}(x)$ is the utility to employee j if he takes the ith job at salary x. Finally, the reservation utility r_i is the utility to i of not hiring anyone, while s_j is j's valuation of the prospect of remaining unemployed. We assume r_i and s_j are finite.

Next, consider a coalition S consisting of an employer i and an employee j. If i hires j, the set of feasible payoffs for S can be given by:

$$W(S) = \{(u_{ij}(-x), v_{ij}(x)), x \epsilon \Re\},$$

[1] For a thorough description of the model and its structure, the reader is advised to see Demange-Gale (1985).

[2] The model here extends without loss of generality to the case where there are m employers and n employees, $m \neq n$, because we can add "dummy" employers [i.e., those whose jobs would never interest any employee—mathematically $v_{ij}(-f_{ij}(r_i)) < s_j \ \forall j$] or "dummy" employees [those who would never be hired] to the model.

where f_{ij} and g_{ij} are the inverse functions of u_{ij} and v_{ij} respectively.[3] On the other hand, if i does not hire j, $(u_i, v_j) = (r_i, s_j)$ is attained. Hence, the possibilities for S can be given by the characteristic function

$$V(S) = \{(u_i, v_j) : f_{ij}(u_i) + g_{ij}(v_j) = 0 \text{ or } (u_i, v_j) = (r_i, s_j)\}. \tag{2.1}$$

At this point, it is useful to make some definitions and notations. First, if i and j do indeed form a coalition, we say that they are matched. An assignment, denoted by μ, is a sequence of matched pairs $\{(i_1\ j_1), ..., (i_m\ j_m)\}$ in which no agent appears more than once. If $(i\ j) \epsilon \mu$, we write $j = \mu(i)$ or $i = \mu^{-1}(j)$.

The core is defined as the set of economic allocations under which it is impossible for any coalition on its own to improve the lot of each of its constituents. Since the essential coalitions of this game are those consisting of a single employer and worker, we can write that the core is the set of assignments μ, employers' utilities $(u_1, ..., u_n)$, and employees' utilities $(v_1, ..., v_n)$ satisfying:

$$f_{ij}(u_i) + g_{ij}(v_j) = 0 \text{ for } j = \mu(i) \tag{2.2}$$

$$u_i = r_i \text{ for } i \text{ unmatched by } \mu \tag{2.3}$$

$$v_j = s_j \text{ for } j \text{ unmatched by } \mu \tag{2.4}$$

$$u_i \geq r_i \text{ for all } i \epsilon I \tag{2.5}$$

$$v_j \geq s_j \text{ for all } j \epsilon J \tag{2.6}$$

$$(u_i, v_j) \not< (u_{ij}(-x), v_{ij}(x)) \ \forall x \epsilon \Re, j \neq \mu(i) \tag{2.7}$$

[3] The representation of $W(S)$ is easier to see once we interpret $f_{ij}(u_i)$ as the amount of money needed to be paid to i in order to raise his welfare to u_i, given that he must be matched with j. Similarly, $g_{ij}(v_j)$ is the amount of money needed to be paid to j in order to raise his welfare to v_j, given that he must be matched to i. These interpetations also will help in understanding the stability constraints (2.7′) for the core.

Note that constraints (2.3)-(2.4) entertain the possibility that i or j be unmatched by μ, i.e., [unlike in Shapley-Shubik (1972)], a μ in the core need not contain n pairs. For this reason, we introduce the following notation: $\mu(i) = \emptyset$ means i is unmatched under μ, while $\mu^{-1}(j) = \emptyset$ indicates the same is true for j.

At any rate, constraints (2.2)-(2.4) represent what the coalitions $[(i\ \mu(i))]$, $[(i)$ where i is unmatched], and $[(j)$ where j is unmatched] can achieve for themselves. Inequalities (2.5) and (2.6) depict individual rationality. In general, we refer to constraints (2.2)-(2.6) as the feasibility constraints because they describe limitations on the welfare of an agent due to the coalition he is in.

In relation (2.7), the symbol "$\not<$" is defined by

$$(a,b) \not< (c,d) \Longleftrightarrow a > c \text{ or } b > d \text{ or } (a,b) = (c,d).$$

Thus, (2.7) means that if i and j are not matched, it is impossible for them to improve both their positions by forming a coalition. Hence we call (2.7) the stability constraints. Note that they can be rewritten as

$$f_{ij}(u_i) + g_{ij}(v_j) \geq 0 \text{ for } j \neq \mu(i). \tag{2.7'}$$

At this point, one should note an important instance of this game. Suppose $u_{ij}(x) = a_{ij} + x$, $v_{ij}(x) = b_{ij} + x$, $r_i = 0$, and $s_j = 0$ for all i and j, where $a_{ij} \geq 0$ and $b_{ij} \geq 0$ are constants. Constraints (2.2)-(2.7') become

$$u_i + v_{\mu(i)} = c_{i\mu(i)}$$

$$u_i + v_j \geq c_{ij}$$

$$u_i, v_j \geq 0\ \forall i, j$$

where $c_{ij} = a_{ij} + b_{ij}\ \forall i, j$. This is precisely the core for Shapley and Shubik's (1972) housing market with "assignment matrix" $C \equiv \{c_{ij}\}$.

It is useful to know which properties of the Shapley-Shubik market extend to the general Gale-Demange setting. First,

Theorem 2.1: *The core is nonempty.*

Proof: Kaneko (1982), Quinzii (1984), or Quint (1987).

Next, we represent the core as a region in the space of employers' utility levels. Define a core assignment as any assignment μ under which there exist $(\underline{u}, \underline{v})$ with $(\mu, \underline{u}, \underline{v})$ in the core. Then, for any core assignment μ, the vector $\underline{u}$ is in the core if

$$u_i = r_i \text{ for } i \text{ unmatched by } \mu \tag{2.3}$$

$$u_i \geq r_i \text{ for all } i \epsilon I \tag{2.5}$$

$$v_{\mu^{-1}(j)j}[-f_{\mu^{-1}(j)j}(u_{\mu^{-1}(j)})] \geq s_j \text{ for all } j \text{ matched by } \mu \tag{2.6'}$$

$$f_{ij}(u_i) + g_{ij}(s_j) \geq 0 \text{ for all } j \text{ unmatched by } \mu \tag{2.7''}$$

$$f_{ij}(u_i) + g_{ij}(v_{\mu^{-1}(j)j}[-f_{\mu^{-1}(j)j}(u_{\mu^{-1}(j)})]) \geq 0 \text{ for all } j \text{ matched by } \mu \tag{2.7'''}$$

Since the functions f_{ij}, g_{ij}, and v_{ij} are all increasing, the relations (2.3)-(2.7''') all depict "i-increasing, $\mu^{-1}(j)$-decreasing" sets. Hence, the set of $\underline{u}$'s satisfying these constraints is a sublattice.[4] Denote this set by U_μ.

Another set that we wish to consider is $U = \cup_{\text{core } \mu\text{'s}} U_\mu$. This represents the set of possible employers' utility vectors if restricted to outcomes in the core. As will be shortly demonstrated, the above union can in general be "nontrivial", i.e., two or more core assignments exist with all U_μ's proper subsets of U. Hence, given $\underline{u} \epsilon U$, it is a nontrivial problem to find a core assignment under which $\underline{u}$ is feasible.

Even though the union of sublattices is in general *not* a sublattice, we have:

Theorem 2.2: *U is a sublattice.*

Proof: Demange and Gale (1985).

[4] See Veinott, Section 2.4, "Finite Meet Representation of Sublattices".

Let $\underline{u}^H$ and $\underline{u}^L$ be the greatest ("employer optimal") and least ("worker optimal") points, respectively, of U. [These points exist because U is a compact sublattice.[5]] Demange and Gale proved the following facts concerning these points:

1) Suppose a subset I' of employers is allowed to falsify the values $\{f_{i'j}(x)\}_{i'\epsilon I'}$ and $\{r_{i'}\}_{i'\epsilon I'}$, say to $\{\tilde{f}_{i'j}(x)\}_{i'\epsilon I'}$ and $\{\tilde{r}_{i'}\}_{i'\epsilon I'}$. Then, if $(\tilde{\mu}, \underline{\tilde{u}}, \underline{\tilde{v}})$ is in the core of this "manipulated" market, then $u_{i'\tilde{\mu}(i')}(-\tilde{x}_{i'}) \leq u^H_{i'}$ for some $i'\epsilon I'$.

2) Suppose the rules of the game are as follows: Any subset J' of workers are allowed to falsify the values $\{g_{ij'}(x)\}_{j'\epsilon J'}$ and $\{s_{j'}\}_{j'\epsilon J'}$. However, given this manipulated market, the *employer optimal* payoff is enforced. Then $\underline{u}^L$ (i.e., $\underline{v}^H$)[6] is a strong equilibrium, i.e., $v_{\mu^{-1}(j')j'}(\tilde{x}_{\mu^{-1}(j')}) \leq v^H_{j'}$ for some $j'\epsilon J'$.

Thus, in these senses, the points $\underline{u}^H$ and $\underline{u}^L$ are significant in that they are nonmanipulable.

At this point, it is useful to present an example in order to demonstrate some of these results.

	j=1	j=2	r_i
i=1	$(x+1)^3$ x	$x-1$ $1+x+\lvert x\rvert^+$	-1
i=2	$(x+1)/2$ $x+3$	x $x+2$	1
s_j	-4	0	

In each cell, the top function is $u_{ij}(x)$ and the bottom function is $v_{ij}(x)$. Again, the r_i's and s_j's are the reservation utilities for the employers and employees.

Consider the matching $\mu^1 = \{(1\ 1)(2\ 2)\}$. We want to investigate whether or not μ^1 is a core assignment. To this end, let α_1 be the salary paid by employer

[5] Veinott, Section 1.2, "Partially Ordered Sets"

[6] Using the same logic as above, we could define the sets V_μ and V, again both of which are sublattices. Its easy to see (Roth 1984) $\underline{u}^H$ and $\underline{v}^L$ always occur simultaneously, i.e., $(\underline{u}^H, \underline{v}^L, \mu)$ satisfies (2.2)-(2.7) iff $(\underline{u}^H, \mu)$ satisfies (2.3)-(2.7'''').

$i = 1$ (to employee $j = 1$), and define α_2 similarly. Then $\underline{u} = ([1-\alpha_1]^3, -\alpha_2)$ and $\underline{v} = (\alpha_1, 2+\alpha_2)$. In order for $(\underline{u}, \underline{v})$ to be feasible, we need $u_i \geq r_i$ and $v_j \geq s_j$ for every i and j. This in turn implies

$$-4 \leq \alpha_1 \leq 2 \tag{2.8}$$

$$-2 \leq \alpha_2 \leq -1 \tag{2.9}$$

To consider the stability constraints, we first need to calculate the inverse functions f_{ij} and g_{ij}:

	j=1	j=2
i=1	$\sqrt[3]{x}-1$ x	$x+1$ $x-\frac{[x-1]^+}{2}-1$
i=2	$2x-1$ $x-3$	x $x-2$

In light of the fact that $2+\alpha_2 \leq 1$, $f_{12}(u_1) + g_{12}(v_2) \geq 0$ can be written as

$$(1-\alpha_1)^3 + 2 + \alpha_2 \geq 0 \tag{2.10}$$

Furthermore, $f_{21}(u_2) + g_{21}(v_1) \geq 0$ is equivalent to

$$\alpha_1 - 2\alpha_2 - 4 \geq 0. \tag{2.11}$$

The set of "core α's", which is the set of solutions to (2.8)-(2.11), is shown in Figure 1. Each of these corresponds to a point in "employer-space"—thus this set (Figure 2) is contained in U.

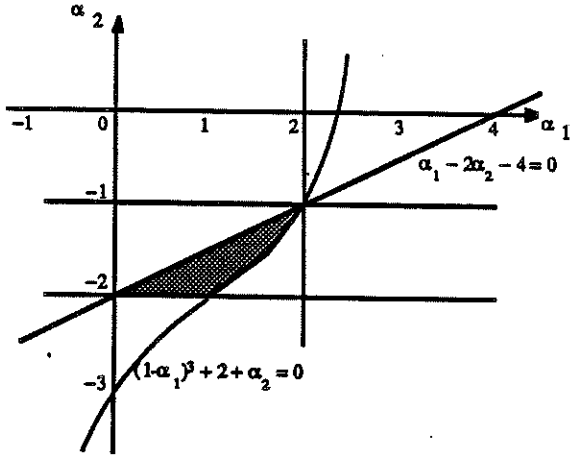

Figure 5

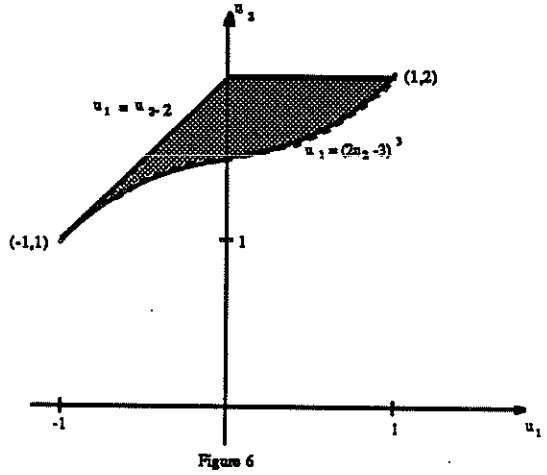

Figure 6

However, the region depicted in Figure 2 is just U_{μ^1}. It is *not* U. This is because, unlike in the Shapley-Shubik case, there is in general more than one core assignment. To wit, in this example, $\mu^2 = \{(1\ 2), (2\ 1)\}$ is another one. Indeed, applying the same procedure with μ^2 yields

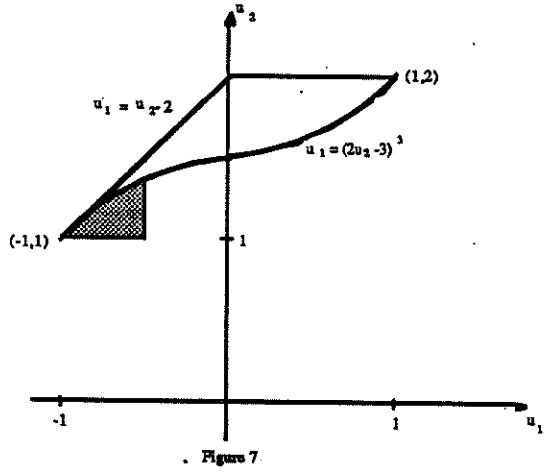

Figure 7

Running through the other six possible assignments, we find that none of them are core assignments. Thus, U is the union of the areas in Figure 2 and Figure 3. Finally, note that $\underline{u}^H = (1, 2)$ is feasible under μ^1 but not μ^2, but that $\underline{u}^L = (-1, 1)$ is attainable under both core assignments.

3. Crawford and Knoer's Algorithm

In this section, we describe Crawford and Knoer's algorithm for finding a point in the core. Given the notation of Section 2, the algorithm is:

1) Choose a "small enough" $\Delta > 0$. Set $t = 0$. Set $\alpha_{ij}(0) = g_{ij}(s_j)\ \forall i, j$.

2) Employer i makes an offer to his favorite worker j, keeping in mind the present salary structure $[\alpha_{i1}(t), ..., \alpha_{in}(t)]$. Employers may break ties however they like.

3) Each worker j who has one or more (unrejected) offer tentatively accepts his favorite and rejects all others. If no one rejects an offer, STOP.

4) Offers not rejected previously remain in effect. If worker j has rejected an offer from firm i in period t, $\alpha_{ij}(t+1) = \alpha_{ij}(t) + \Delta$. If not, $\alpha_{ij}(t+1) = \alpha_{ij}(t)$.

5) Set $t = t + 1$ and return to 2).

A few comments are in order here. First, it should be noted that this procedure is an analogue for the well-known Gale-Shapley algorithm (which solves the problem in the case where the utility functions $u_{ij}(x)$ and $v_{ij}(x)$ are replaced by ordinal preferences).[7] For instance, in the Gale-Shapley algorithm, each employer begins by making the most selfish offer possible—he proposes to his favorite worker. Likewise, in Crawford and Knoer, the employers initially consider paying employees just enough to bring them to their reservation utilities (subsistence wages?!), and then, *given this advantageous salary structure, they choose their favorite workers.*

As both algorithms progress, prospects for the employers grow dimmer, while for the workers, they grow brighter. In the Gale-Shapley algorithm, this follows from the fact that in every round, employers either don't make a new offer, or make one to a worker one notch lower in the firm's preference ordering. In the Crawford-Knoer procedure, it is evident from the observation that the "permissible salary offers" α_{ij} always either stay constant or go up by Δ.

Finally, there is the relationship between the output of these algorithms and employer-optimality. Assuming that an employer is never indifferent in choosing

[7] See Gale and Shapley (1962).

between two workers (and vice versa), the Gale-Shapley algorithm terminates at an employer-optimal core element. (The core is a [sub]lattice in the Gale-Shapley setup also.) To see the analogous result with Crawford-Knoer, consider the "discretized" market, where firms are only allowed to offer salaries of the form $g_{ij}(s_j)+k\Delta$, where k is an integer. Then, if no firm or worker is ever indifferent between potential partners at any permitted salary, Crawford and Knoer's algorithm terminates at the firm-optimal utilities (for the discretized game).

In all three of these respects, our algorithm, described in the next section, is similar to those of Gale & Shapley and Crawford & Knoer.

However, there is one aspect of the Crawford-Knoer procedure which we aim to improve. This concerns the "price adjustment factor" Δ. Crawford and Knoer's idea for finding a core point is to set Δ small enough so that the discretized game closely approximates the real, "continuous" game. As an example, Crawford has suggested that a Δ less than 1 is necessary for the algorithm to work for general Shapley-Shubik games where all c_{ij}'s are integers.[8]

However, the smaller Δ is, the longer the algorithm takes to run, especially if core salaries are high. For example, consider the following example

	j=1	j=2	r_i
i=1	600+x 400+x	x x	0
i=2	600+x 401+x	x x	0
s_j	0	0	

Using Crawford-Knoer with $\Delta = 1$, we initialize at $\alpha_{11}(0) = -400$, $\alpha_{21}(0) = -401$, $\alpha_{12}(0) = \alpha_{21}(0) = 0$. Two thousand iterations later, the algorithm ends, with $\alpha_{11}(2000) = 600$, $\alpha_{21}(2000) = 599$, $\alpha_{12}(2000) = 0$, $\alpha_{22}(2000) = 0$, $\mu = \{(1\ 2)(2\ 1)\}$, $\underline{u} = (0, 1)$, and $\underline{v} = (1000, 0)$. Clearly we would like an algorithm whose running

[8] Personal correspondence with Crawford, 1987.

time is not dependent on what the final salaries are. Put another way, since each of the 2000 iterations involve both employers making an offer to worker 1,we would like an algorithm which would roll all of these steps up into one. This is the main thrust of the next section.

4. Our Algorithm

To describe our algorithm, we will use a mathematical approach. As we proceed, economic interpretations will be supplied.

So suppose we're given a market defined by $\{u_{ij}(x), v_{ij}(x), r_i, s_j\}_{i,j=1}^n$. Given an assignment μ, define the variables $\{p_{ij}\}$ by:

$$p_{ij} = \begin{cases} 1; & \text{if } \mu(i) = j; \\ 0 & \text{otherwise.} \end{cases}$$

Using this notation, the core elements [i.e., $(\underline{u}, \underline{v}, \mu)$ satisfying (2.2)-(2.7)] become $(\underline{u}, \underline{v}, \underline{p})$ solving

$$f_{ij}(u_i) + g_{ij}(v_j) = 0 \text{ if } p_{ij} = 1 \tag{4.1}$$

$$u_i = r_i \text{ if } p_{ij} = 0 \ \forall \ j \tag{4.2}$$

$$v_j = s_j \text{ if } p_{ij} = 0 \ \forall \ i \tag{4.3}$$

$$u_i \geq r_i \text{ for all } i \epsilon I \tag{4.4}$$

$$v_j \geq s_j \text{ for all } j \epsilon J \tag{4.5}$$

$$f_{ij}(u_i) + g_{ij}(v_j) \geq 0 \text{ for all } i, j \tag{4.6}$$

$$\sum_{j=1}^{n} p_{ij} \leq 1 \text{ for all } i \epsilon I \tag{4.7}$$

$$\sum_{i=1}^{n} p_{ij} \leq 1 \text{ for all } j \epsilon J \tag{4.8}$$

$$p_{ij} = 0 \text{ or } 1 \ \forall \ i, j.$$

Statement of Algorithm and Proof of Convergence

Our algorithm again initializes with each employer deciding which employee to offer subsistance wages to.

Definition: Given a game defined by $\{u_{ij}(x), v_{ij}(x), r_i, s_j\}_{i,j=1}^n$, the U-solution is the vector $(\underline{u}, \underline{v}, \underline{p})$ given by:

$$u_i = \begin{cases} \max_j u_{ij}(-g_{ij}(s_j)) = a_i, & \text{if } a_i > r_i; \\ r_i & \text{otherwise.} \end{cases}$$

$$p_{ij} = \begin{cases} 1, & \text{if } j \epsilon \text{ argmax above, } k \notin \text{argmax for } k < j, \text{ and } a_i > r_i; \\ 0 & \text{otherwise.} \end{cases}$$

$$v_j = s_j \ \forall \ i, j$$

The interpretation here is that, for each employer i, $p_{ij} = 1$ means that he decides to make an initial offer to worker j.

Lemma 1: *The U-solution satisfies constraints (4.1)-(4.7).*

At this point, we are able to describe the idea of the algorithm. We start with the U-solution. The iterated step is a pivoting procedure which preserves (4.1)-(4.7), and which, if repeated enough times, eventually leads to a vector $(\underline{u}, \underline{v}, \underline{p})$ which satisfies (4.8) as well.

With this in mind, we now present the entire algorithm formally:

1) Set $(\underline{u}, \underline{v}, \underline{p})$ equal to the U-solution. Set $N = 1$.

2) Set $K(j) = \{i : p_{ij} = 1\}$ for all j. If $|K(j)| \leq 1$ for all j, STOP—the vector $\lim_{\omega \to 0}(\underline{u}, \underline{v}, \underline{p})$ is in the core.

3) Let $J^* = \{j : |K(j)| \geq 2\}$. For each $j^* \epsilon J^*$, do the following:

A) For each $i \epsilon K(j^*)$, calculate

$$a_i = \max_{j \neq j^*} u_{ij}(-g_{ij}(v_j))$$

$$\hat{j}(i) = \begin{cases} \text{argmax above,} & \text{if } a_i > r_i; \\ \emptyset & \text{otherwise.} \end{cases}$$

If |argmax| ≥ 2, just choose $\hat{j}$ arbitrarily. Set $a_i^* = \max(a_i, r_i)$.

B) Set

$$w_{j^*} = \max_{i \epsilon K(j^*)} v_{ij^*}(-f_{ij^*}(a_i^*))$$

$$\bar{i}(j^*) = \text{argmax above.}$$

If $|\bar{i}(j^*)| > 1$, arbitrarily select $i \epsilon \bar{i}(j^*)$ and change $v_{ij^*}(x)$ to $v_{ij^*}(x) + 2^{-N}\omega$ [ω an infinitesimal]. Let $N = N + 1$, and return to 3B)—now $|\bar{i}(j)|$ should be equal to 1.

4) 'Pivot' on $\bar{j}$, where $\bar{j}$ is a maximizer of $\Delta v_j = w_j - v_j$ over J^*:

A) $v_{\bar{j}}$ changes to $v_{\bar{i}(\bar{j})\bar{j}}(-f_{\bar{i}(\bar{j})\bar{j}}(a^*_{\bar{i}(\bar{j})}))$.

B) $p_{i\bar{j}}$ changes from 1 to 0 for all $i \epsilon K(\bar{j})$ with $i \neq \bar{i}(\bar{j})$.

C) $p_{i\hat{j}(i)}$ changes from 0 to 1 for all $i \epsilon K(\bar{j})$ with $i \neq \bar{i}(\bar{j})$.[9]

D) u_i changes to a_i^* for all $i \epsilon K(\bar{j})$.

Return to 2).

Again, Step 1 is the initialization step, where the offers described by the U-solution are made (see above).

Step 2 gives the condition for the algorithm to end, namely that inequality (4.8) is satisfied.

Steps 3 and 4 describe how to move from one prospective solution to another. Consider any employee j^* with two or more offers. $K(j^*)$ is the set of employers who are currently making offers to j^*. The firms in $K(j^*)$ now have a "bidding war"—each attempts to win j^*'s services for itself by raising j^*'s tentative salary. How much of a raise is employer i willing to give? The answer depends upon his next best alternative. Suppose i is barred from dealing with j^*. Then a_i represents his highest attainable utility through forming a coalition with another employee j, provided he must "match all offers" to j. Since he also has the option of not making

[9] Of course, if $\hat{j}(i) = \emptyset$, this step is ignored for i.

an offer at all (which pays off r_i), a_i becomes a_i^*. Thus, i is willing to increase j^*'s wage as long as $u_i \geq a_i^*$. In other words, he is willing to pay a salary of $-f_{ij^*}(a_i^*)$ in order to "keep" j^*.

To j^*, the best of these offers naturally maximizes $v_{ij^*}(-f_{ij^*}(a_i^*))$, yielding w_{j^*}. However, the bidding war for j^* does not necessarily end at this point—in general, it continues on into the next iteration. This is because only one bidding war is resolved per iteration—and that is the one for $\overline{j}$, the employee in J^* for whom $w_j - v_j$ is maximum. Worker $\overline{j}$ agrees to reject all offers other than the best one [from $\overline{i}(\overline{j})$]. In return, $\overline{i}$ raises $\overline{j}$'s salary offer, from $-f_{\overline{i}(\overline{j})\overline{j}}(u_{\overline{i}})$ to $-f_{\overline{i}(\overline{j})\overline{j}}(a_{\overline{i}}^*)$. As for the rejected employers, i.e., all those in $K(\overline{j})$ other than $\overline{i}(\overline{j})$, they now pursue their next best alternative. The process repeats.

Before proceeding further, the reader should note that the algorithm contains a "degeneracy" procedure to handle cases where ties occur in the selection of $\overline{i}(j)$ [see the end of step 3B)]. In a nutshell, the ω's serve to lexicographically order the $v_{ij}(x)$'s among the employers $i \epsilon I$. This prevents the algorithm from cycling in certain cases, and thus enables the procedure to work on a larger set of problem instances. In particular, see Theorem 4.3.

Next, let us adopt the following convention. For any of the quantities defined in the algorithm, affix a superscript "t" to refer to the quantity in the tth iteration. Hence, u_i^t means u_i during the tth iteration, a_i^{*t} means i's next best alternative during the tth iteration, etc.

Lemma 2: *Let* $(\underline{u}^t, \underline{v}^t, \underline{p}^t)$ *solve constraints (4.1)-(4.7). Let* $(\underline{u}^{t+1}, \underline{v}^{t+1}, \underline{p}^{t+1})$ *result from applying one iteration of the algorithm to* $(\underline{u}^t, \underline{v}^t, \underline{p}^t)$. *Then* $\underline{v}^t \leq \underline{v}^{t+1}$.

Lemma 2 formally states what we found to be true in the Gale-Shapley and Crawford-Knoer cases, namely, that as the algorithm progresses, the welfares of the employees increase (and those of the firms decrease).

Proof: We begin with a claim, which simply states that for any employer, his next best alternative is worse than his current state.

Claim: $u_i^t \geq a_i^{*t}$.

Proof: Suppose not. Then $a_i^{*t} = \max[r_i, u_{i\hat{j}(i)}(-g_{i\hat{j}(i)}(v_{\hat{j}(i)}^t))] > u_i^t$. Now $r_i > u_i^t$ is impossible because $(\underline{u}^t, \underline{v}^t, \underline{p}^t)$ satisfies (4.4), so it must be that

$$u_{i\hat{j}(i)}(-g_{i\hat{j}(i)}(v_{\hat{j}(i)}^t)) > u_i^t, \text{ or}$$

$$f_{i\hat{j}(i)}(u_i^t) + g_{i\hat{j}(i)}(v_{\hat{j}}^t) < 0.$$

But this contradicts (4.6) for $(\underline{u}^t, \underline{v}^t, \underline{p}^t)$.

Now Lemma 2 is easily proven. For $j \neq \bar{j}^t$, $v_j^{t+1} = v_j^t$ from the definition of the algorithm. Since $v_{ij}(x)$ and $f_{ij}(x)$ are increasing,

$$v_{\bar{j}}^{t+1} = v_{\bar{i}(\bar{j})\bar{j}}(-f_{\bar{i}(\bar{j})\bar{j}}(a_{\bar{i}(j)}^{*t})) \geq v_{\bar{i}(\bar{j})\bar{j}}(-f_{\bar{i}(\bar{j})\bar{j}}(u_{\bar{i}}^t)) = v_{\bar{j}}^t.$$

Lemma 3: *Suppose* $(\underline{u}^t, \underline{v}^t, \underline{p}^t)$ *satisfies (4.1)-(4.7). Let* $(\underline{u}^{t+1}, \underline{v}^{t+1}, \underline{p}^{t+1})$ *result from applying one iteration of the algorithm to* $(\underline{u}^t, \underline{v}^t, \underline{p}^t)$. *Then* $(\underline{u}^{t+1}, \underline{v}^{t+1}, \underline{p}^{t+1})$ *satisfies (4.1)-(4.7) as well.*

Proof: From the definition of the algorithm, it is clear that constraints (4.2)-(4.5) and (4.7) must hold for $(\underline{u}^{t+1}, \underline{v}^{t+1}, \underline{p}^{t+1})$. To show (4.1) and (4.6) are satisfied as well, consider the following cases:

Case 1: $i = \bar{i}(\bar{j}^t)$, $j = \bar{j}^t$ [so $p_{ij}^{t+1} = 1$].

Then, $f_{ij}(u_i^{t+1}) + g_{ij}(v_j^{t+1}) = 0$ from the definitions of $u_{\bar{i}(\bar{j})}$ and $v_{\bar{j}}$ presented in Step 4 of the algorithm.

Case 2: $i = \bar{i}(\bar{j}^t)$, $j \neq \bar{j}^t$ [$p_{ij}^{t+1} = 0$]. In this case,

$$\begin{aligned} f_{ij}(u_i^{t+1}) + g_{ij}(v_j^{t+1}) &\geq f_{ij}(\max_{k \neq \bar{j}} u_{ik}(-g_{ik}(v_k^t))) + g_{ij}(v_j^t) \\ &\geq f_{ij}(u_{ij}(-g_{ij}(v_j^t))) + g_{ij}(v_j^t) = 0. \end{aligned}$$

Case 3: $i \epsilon K(\overline{j}^t))$ but $i \neq \overline{i}(\overline{j}^t)$, $j = \overline{j}^t$ $[p_{ij}^{t+1} = 0]$.

$$f_{ij}(u_i^{t+1}) + g_{ij}(v_j^{t+1}) = f_{ij}(a_i^{*t}) + g_{ij}(\max_{i \epsilon K(\overline{j}^t)} v_{ij}(-f_{ij}(a_i^{*t})))$$
$$\geq f_{ij}(a_i^{*t}) + g_{ij}(v_{ij}(-f_{ij}(a_i^{*t}))) = 0$$

Case 4: $i \epsilon K(\overline{j}^t)$ but $i \neq \overline{i}(\overline{j}^t)$, $j = \hat{j}(i)$ $[p_{ij}^{t+1} = 1]$.

$$f_{ij}(u_i^{t+1}) + g_{ij}(v_j^{t+1}) = f_{i\hat{j}(i)}(a_i^{*t}) + g_{i\hat{j}(i)}(v_{\hat{j}(i)}^t)$$
$$= f_{i\hat{j}(i)}(u_{i\hat{j}(i)}(-g_{i\hat{j}(i)}(v_j^t))) + g_{u\hat{j}(i)}(v_{\hat{j}(i)}^t) = 0.$$

Case 5: $i \epsilon K(\overline{j}^t)$ but $i \neq \overline{i}(\overline{j}^t)$, $j \neq \hat{j}(i)$, $j \neq \overline{j}^t$ $[p_{ij}^{t+1} = 0]$.

$$f_{ij}(u_i^{t+1}) + g_{ij}(v_j^{t+1}) = f_{ij}(a_i^{*t}) + g_{ij}(v_j^t)$$
$$= f_{ij}(\max_{k \neq \overline{j}} u_{ik}(-g_{ik}(v_k^t))) + g_{ij}(v_j^t)$$
$$\geq f_{ij}(u_{ij}(-g_{ij}(v_j^t))) + g_{ij}(v_j^t) = 0.$$

Case 6: $i \notin K(\overline{j}^t)$, $j = \overline{j}^t$ $[p_{ij}^{t+1} = 0]$.

$$f_{ij}(u_i^{t+1}) + g_{ij}(v_j^{t+1}) \geq f_{ij}(u_i^t) + g_{ij}(v_j^t) \geq 0,$$

where the first inequality holds by Lemma 2.

Case 7: $i \notin K(\overline{j}^t)$, $j \neq \overline{j}^t$.

In this case $u_i^{t+1} = u_i^t$, $v_j^{t+1} = v_j^t$, and $p_{ij}^{t+1} = p_{ij}^t$. So, since (4.1) and (4.6) hold at the tth iteration, they also hold at the $t+1$th.

Summarizing, we initialize the algorithm with the U-solution, and pivot and pivot, always preserving (4.1)-(4.7). But how do we know that eventually (4.8) will be solved? The idea is hinted at by Lemma 2. Since $\underline{v}^t$ is increasing it t, $\underline{u}^t$ is decreasing. If enough iterations are made, the u_i's will eventually start to dip below the corresponding r_i's. Then the employers will not be making offers (see step 3A of the algorithm) and thus the chances that $|K(j)| \leq 1$ for all j will inexorably increase.

Theorem 4.1: *Suppose* $\exists \delta > 0$ *such that, for all* t, *there is a finite* $T > t$ *where*

$$\exists i \epsilon K(\bar{j}^T) \text{ with } -\Delta u_i \equiv u_i^T - u_i^{T+1} = u_i^T - a_i^{*T} \geq \delta \tag{4.9}$$

Then the algorithm terminates in a finite number of steps at an element in the core.

Proof: Suppose not. Then, suppose t is a time at which there have been $\tau = 1 + (1/\delta) * \sum_{i=1}^{n} [\max_j u_{ij}(-g_{ij}(s_j)) - r_i]^+$ steps satisfying (4.9). We have $\sum_{i=1}^{n}(u_i^0 - u_i^t) \geq \delta\tau$. Since $u_i^0 = \max_j u_{ij}(-g_{ij}(s_j))\ \forall i$, it must be that for some i, $u_i^t < r_i$. But this is a contradiction of Lemma 3.

In practice, the condition presented in Theorem 4.1 is quite mild. In fact, it is weaker than the "no ties" assumption usually made[10]—"no ties" can be interpreted as condition (4.9) holding on *every* iteration. For the case of the Shapley-Shubik market, we shall see (Theorem 4.3) that the condition of Theorem 4.1 is satisfied whenever the associated assignment linear program is nondegenerate.

Employer-optimality of Output

Theorem 4.2: *Suppose the algorithm converges, say, to* $(\underline{u}^*, \underline{v}^*, \mu^*)$. *Then* $\underline{u}^H \epsilon U_{\mu^*}$.

Suppose we ran the algorithm on the example presented at the end of section 2. Then Theorem 4.2 implies that the output will lie within the shaded region of Figure 2 and *not* the shaded region of Figure 3.

Proof: We begin with a lemma:

Lemma: *Let* $(\underline{u}^t, \underline{v}^t, \underline{p}^t)$ *be the vector generated at the* t*th iteration of the algorithm,* t *arbitrary. Let* $(\hat{\underline{u}}, \hat{\underline{v}}, \hat{\mu})$ *be any element in the core. Then, for any* j,

$$\text{either } \hat{v}_j \geq v_j^t \text{ or } \hat{\mu}^{-1}(j) \epsilon K^t(j). \tag{4.10}$$

[10] See for instance, Gale-Shapley (1962) or Crawford-Knoer (1981).

Proof: It is easy to see that this property holds for $t = 0$, because $v_j^0 = s_j$ for all j. So assume it true for iteration t, and attempt to prove it for iteration $t+1$.

Case 1: $j \neq \overline{j}^t$.

Then $v_j^{t+1} = v_j^t$, and $K^t(j) \subseteq K^{t+1}(j)$. So (4.10) holds by inductive hypothesis.

Case 2: $j = \overline{j}^t$.

Proposition: *Suppose (4.10) holds for some t. Then, for all i,*

$$\text{either } a_i^{*t} \geq \hat{u}_i \text{ or } \mu^t(i) = \hat{\mu}(i)$$

[Here the notation $\mu^t(i) = j$ means $p_{ij}^t = 1$.]

Proof: Suppose not. Then it must be that $a_{\tilde{i}}^{*t} < \hat{u}_{\tilde{i}}$ and $\hat{\mu}(\tilde{i}) = \tilde{j} \neq \mu^t(\tilde{i})$ for some $\tilde{i}$. Since $(\underline{\hat{u}}, \underline{\hat{v}}, \hat{\mu})$ satisfies (4.1),

$$f_{\tilde{i}\tilde{j}}(\hat{u}_{\tilde{i}}) + g_{\tilde{i}\tilde{j}}(\hat{v}_{\tilde{j}}) = 0 \Longrightarrow f_{\tilde{i}\tilde{j}}(a_{\tilde{i}}^{*t}) + g_{\tilde{i}\tilde{j}}(\hat{v}_{\tilde{j}}) < 0 \Longrightarrow a_{\tilde{i}}^{*t} < u_{\tilde{i}\tilde{j}}(-g_{\tilde{i}\tilde{j}}(\hat{v}_{\tilde{j}})).$$

By definition of $a_{\tilde{i}}^{*t}$, $a_{\tilde{i}}^{*t} \geq u_{\tilde{i}\tilde{j}}(-g_{\tilde{i}\tilde{j}}(v_{\tilde{j}}^t))$, so we have $\hat{v}_{\tilde{j}} < v_{\tilde{j}}^t$. But this contradicts (4.10), because $\tilde{j} \neq \mu^t(\tilde{i})$ means $\tilde{i} \notin K^t(\tilde{j})$.

With the Proposition in hand, we can now prove the Lemma for Case 2 above.

Again, assume the contrapositive, i.e.,

$$\hat{v}_{\overline{j}} < v_{\overline{j}}^{t+1} \text{ and } \hat{\mu}^{-1}(\overline{j}) \notin K^{t+1}(\overline{j}).$$

From the definition of the pivoting procedure, $K^{t+1}(\overline{j}) = \{\overline{i}(\overline{j})\}$. Since $u_{\overline{i}}^{t+1} = a_{\overline{i}}^{*t}$,

$$f_{\overline{i}(\overline{j})\overline{j}}(a_{\overline{i}}^{*t}) + g_{\overline{i}(\overline{j})\overline{j}}(v_{\overline{j}}^{t+1}) = 0.$$

This implies

$$f_{\overline{i}(\overline{j})\overline{j}}(a_{\overline{i}}^{*t}) + g_{\overline{i}(\overline{j})\overline{j}}(\hat{v}_{\overline{j}}) < 0,$$

which further implies $a_{\overline{i}}^{*t} < \hat{u}_{\overline{i}}$, because $(\underline{\hat{u}}, \underline{\hat{v}}, \hat{\mu})$ is in the core. But also, $K^{t+1}(\overline{j}) = \{\overline{i}(\overline{j})\} \Longrightarrow \hat{\mu}^{-1}(\overline{j}) \neq \overline{i} \Longrightarrow \hat{\mu}(\overline{i}) \neq \overline{j} = \mu^t(\overline{i})$, so the Proposition is contradicted.

Corollary: *Let* $(\underline{u}^t, \underline{v}^t, \underline{p}^t)$ *be the vector generated at the tth iteration of the algorithm, t arbitrary. Let* $(\underline{\hat{u}}, \underline{\hat{v}}, \hat{\mu})$ *be any element in the core. Then, for any* i,

$$\text{either } a_i^{*t} \geq \hat{u}_i \text{ or } \mu^t(i) = \hat{\mu}(i).$$

Proof: Follows directly from the Lemma and the Proposition.

At this point, we are ready to prove the Theorem. Let $(\underline{u}^H, \underline{v}^L, \mu^H)$ be the employer-optimal point of the core, and let $(\underline{u}^*, \underline{v}^*, \mu^*)$ be the output of the algorithm. We need to show that $(\underline{u}^H, \underline{v}^L, \mu^*)$ is in the core.

Define the sets

$$\begin{aligned}
I_1 &= \{i : u_i^H > u_i^*\}, \\
J_1 &= \{j : \mu^H(i) = j,\ i \epsilon I_1\} \\
&= \{j : \mu^*(i) = j,\ i \epsilon I_1\} \text{ because of the Corollary,} \\
I_2 &= \{i : u_i^H = u_i^*\} = I_1^c, \\
J_2 &= J_1^c.
\end{aligned}$$

Then, by the corollary to Lemma 1 in Demange-Gale (1985), we have

$$v_j^L = v_j^* \text{ for all } j \epsilon J_2.$$

To show that $(\underline{u}^H, \underline{v}^L, \mu^*)$ is feasible, we consider two cases:

1) $i \epsilon I_1,\ j \epsilon J_1,\ j = \mu^*(i)$:

Since $\mu^*(i) = \mu^H(i)$ in this case, the fact that u_i^H, v_j^L is feasible under μ^H means that it is also feasible under μ^*.

2) $i \epsilon I_2, j \epsilon J_2$:

Since $\{\underline{u}^H, \underline{v}^L\}_{i \epsilon I_2, j \epsilon J_2} = \{\underline{u}^*, \underline{v}^*\}_{i \epsilon I_2, j \epsilon J_2}$ in this case, the fact that $\{\underline{u}^*, \underline{v}^*\}_{i \epsilon I_2, j \epsilon J_2}$ is feasible under μ^* means that $\{\underline{u}^H, \underline{v}^L\}_{i \epsilon I_2, j \epsilon J_2}$ is feasible under μ^*.

Last, the stability of a triple $(\underline{u}, \underline{v}, \mu)$ does not depend on μ; hence $(\underline{u}^H, \underline{v}^L, \mu^H)$'s stability implies that $(\underline{u}^H, \underline{v}^L, \mu^*)$ is stable as well.

Theorem 4.2 expresses the sense in which the algorithm's output is "employer-optimal" (see p. 11). Vector $(\underline{u}^*, \underline{v}^*)$ is not in general employer-optimal; however, in a sense, assignment μ^* is. At any rate, having run the algorithm and found μ^*, the problem of finding $\underline{u}^H$ is reduced to a "simple" non-linear program. All we need do is maximize an increasing function of $\underline{u}$, say $\sum_{i=1}^{n} u_i$, over the set of constraints (2.3)-(2.7‴) [p. 6], *with μ set equal to μ^**. [11]

Comparison to Crawford & Knoer

It is also interesting to compare our algorithm to that of Crawford and Knoer. The major difference lies in the price adjustment mechanism. As stated in the last section, in Crawford and Knoer's algorithm, salaries always rise by a constant Δ. In our algorithm, the amount of increase is flexible—it depends on what the employers' next best alternatives are. This enables our algorithm to run much faster than Crawford-Knoer in cases where core salaries for the employees are relatively high. For instance, consider the example presented at the end of the previous section. Both algorithms begin with both firms offering worker 1 a salary of zero. With $\Delta = 1$, the Crawford-Knoer algorithm takes 2000 steps to reach a solution in the core with worker 1's wage equal to 1000. On the other hand, our algorithm looks at the U-solution and notices that the next best alternative for both firms is to get zero (instead of 1000 and 1001 respectively). Hence, the first (and only) step of the algorithm raises α_1 by 1000 and results in the same ending core point.

Also, the fact that we don't have to first calculate a "Δ" places our algorithm at a comparative advantage.

However, in general, the Crawford-Knoer algorithm will result in a core point closer to $\underline{u}^H$ than will our algorithm. This is because their algorithm is taking

[11] Without knowledge of μ^*, the constraint set is a system of inequalities containing complementarity constraints. This is the sense in which the NLP of finding $\underline{u}^H$ is simplified.

smaller "steps" in the space of utilities than is ours. Thus, there is more of a tendency to "overshoot" $\underline{v}^L$ (or $\underline{u}^H$) using our algorithm (see Figure 4).

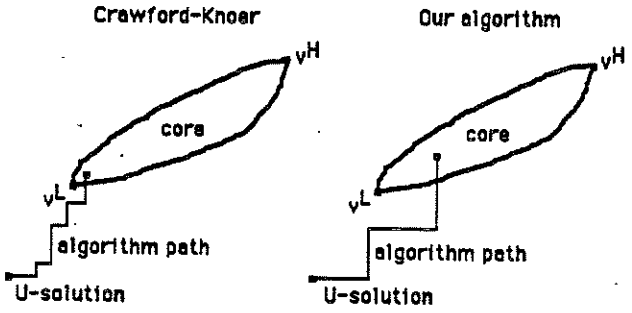

Figure 8

Convergence in the Shapley-Shubik Case

Another issue which Crawford and Knoer consider is the relationship between algorithm convergence and core nondegeneracy. They remark that "We conjecture that...the salary adjustment process may fail to converge to the core of the corresponding continuous market even when that core is nondegenerate". The next and last theorem addresses this question for our algorithm in the case of the Shapley-Shubik market (p. 5).

Definition: Suppose C is an $n \times n$ matrix. Then the assignment linear program associated with C is given by

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} p_{ij} \qquad (P)$$

$$\text{s.t.} \sum_{j=1}^{n} p_{ij} \leq 1$$

$$\sum_{i=1}^{n} p_{ij} \leq 1$$

$$p_{ij} \geq 0 \ \forall i, j$$

Definition: Suppose C is an $n \times n$ matrix. Then C is nondegenerate if the assignment linear program (P) associated with C has a unique solution.

Theorem 4.3: *Let C be the assignment matrix for a Shapley-Shubik market. If*

1) c_{ij} is rational for all i, j, and

2) C is nondegenerate,

the algorithm terminates at a core point in a finite number of steps.

Proof: Suppose $c_{ij} = e_{ij}/d_{ij}$, where e_{ij} and d_{ij} are nonnegative integers with $d_{ij} > 0$ for every i and j. For the Shapley-Shubik case, the quantities u_i^t will be sums and differences of the original c_{ij}'s for every i and t. Thus, if it is nonzero[12], $-\Delta u_i$ will not be less than $1/\prod_{i,j} d_{ij}$ for each i on each iteration. Hence, we need to show that there can't be more than a finite number of consecutive pivots with $-\Delta u_i = 0 \; \forall i$. That way, we will be able to prove Theorem 4.3 by invoking Theorem 4.1 with $\delta = 1/\prod_{i,j} d_{ij}$.

To do this, we need some definitions.

Definition: The reduced matrix C^{tR} is the matrix defined by

$$c_{ij}^{tR} = c_{ij} - v_j^t.$$

Since v_j^t is increasing in t (i.e., increasing over time), C^{tR} represents the matrix of remaining bargainable surplus for the firms at iteration t. With this interpretation, it is easy to see $\mu^t(i) \epsilon \arg\max_j c_{ij}^{tR}$ and $a_i^t = \max_{j \neq \mu^t(i)} c_{ij}^{tR}$. Thus, if $i \epsilon K(\vec{j}^t)$, $-\Delta u_i = c_{i\mu^t(i)}^{tR} - a_i^t$. Thus, in order for $-\Delta u_i$ to be equal to zero, the two maximal elements in row i of C^{tR} must be equal.

Also, we remark that C is nondegenerate iff C^{tR} is nondegenerate for every t.

[12] For now on, when we speak of terms such as $-\Delta u_i$, C^{tR}, etc, we mean $\lim_{\omega \to 0} -\Delta u_i$, $\lim_{\omega \to 0} C^{tR}$, etc.

Definition: A matrix C has the D-condition if $\exists$ a set of rows $\hat{I} = \{i_1, ..., i_\ell\}$ with $|\hat{I}| \geq 1$, a set of columns $\hat{J} = \{j_1, ..., j_m\}$, and a mapping $\sigma : \hat{I}^c \longrightarrow J$ such that:

1) For every $i \epsilon \hat{I}$, the two maximal elements of row i are in columns of $\hat{J}$

2) For every $i \epsilon \hat{I}$, the two maximal elements of row i are equal

3) For any $i_1, i_2 \epsilon \hat{I}^c$,

A) $\sigma(i_1) \epsilon \arg\max_j c_{i_1 j}$

B) $\sigma(i_1) \notin \hat{J}$

C) $\sigma(i_1) \neq \sigma(i_2)$

Remark: Suppose in every row of C, the two maximal elements are equal. Then C has the D-condition with $\hat{I} = \hat{J} = \{1, ..., n\}$.

Example:

$$C = \begin{pmatrix} 2 & 2 & 0 & 1 & 2 & 0 \\ 2 & 4 & 4 & 0 & 0 & 3 \\ 3 & 6 & 6 & 4 & 1 & 2 \\ 5 & 5 & 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 & 4 & 3 \\ 1 & 7 & 2 & 0 & 8 & 5 \end{pmatrix}$$

Here C has the D-condition with $\hat{I} = \{1, 2, 3, 4\}$, $\hat{J} = \{1, 2, 3\}$, $\sigma(5) = 4$, and $\sigma(6) = 5$.

Lemma 1: *If an $n \times n$ matrix has the D-condition, it is degenerate.*

Proof: See Appendix.

The upshot of this Lemma (and a previous remark) is that, for any nondegenerate assignment matrix C, C^{tR} *cannot have the D-condition for any t.*

Proposition: *Let $n = 3$. Suppose we run the algorithm and, starting at time t, obtain an infinite number of consecutive pivots with $-\Delta u_i = 0 \ \forall i$. Then C^{tR} has the D-condition.*

Proof: Clearly $|J^{*t}| = 1$, so, without loss of generality, let $\vec{j}^t = 1$. Also, w.l.o.g.,

suppose $i = 1, 2 \epsilon K(\bar{j}^t)$. Thus, either $c_{11}^{tR} = c_{12}^{tR}$ $[\geq c_{13}^{tR}]$ or $c_{11}^{tR} = c_{13}^{tR}$ $[\geq c_{12}^{tR}]$. Similarly, either $c_{21}^{tR} = c_{22}^{tR}$ $[\geq c_{23}^{tR}]$ or $c_{21}^{tR} = c_{23}^{tR}$ $[\geq c_{22}^{tR}]$.

Case 1: The two maximal elements of row 3 are unequal.

Case 1A: $3 \epsilon K(\bar{j})$ as well.

Then $\bar{i}(\bar{j}^t) = 3$ and $-\Delta u_3 >> 0$, so we have violated the Proposition's hypothesis.

Case 1B: $3 \notin K(\bar{j})$.

The first pivot puts $\mu^{t+1}(1) \neq \mu^{t+1}(2)$, and the second pivot (if in fact the algorithm hasn't terminated) yields $-\Delta u_3 >> 0$ because $3 \epsilon K(\bar{j}^{t+1})$.

Case 2: The maximal two elements of row 3 are equal.

In this case, C^{tR} has the D-condition (with $\hat{I} = \hat{J} = \{1, 2, 3\}$).

Lemma 2: *Suppose we run the algorithm and, starting at time t, obtain an infinite number of consecutive pivots with* $-\Delta u_i = 0 \ \forall i$. *Then* C^{tR} *has the D-condition.*

Note that proving this Lemma will be tantamount to proving Theorem 4.3.

Proof: By induction on n. The last Proposition shows that the Lemma is true for $n = 3$, so suppose its true for $n - 1$.

So assume for the purpose of contradiction that C^{tR} does not have the D-condition. Thus, $\exists \tilde{i}$ such that the two maximal elements in row $\tilde{i}$ of C^{tR} are unequal (see the last Remark). Let $\tilde{j} = \mu^t(\tilde{i})$. Clearly $\mu^\tau(i) \neq \tilde{j}$ for all $\tau \geq t, i \neq \tilde{i}$, because, otherwise, a pivot would ensue on column $\tilde{j}$, yielding $-\Delta u_{\tilde{i}} >> 0$. However, the degeneracy procedure ensures that furthermore, $\tilde{j} \notin \arg\max_j c_{ij}^{tR}$ for all $\tau \geq t, i \neq \tilde{i}$. In other words, $\tilde{j}$ cannot even be a ***next best alternative*** [Again, otherwise a pivot would eventually occur on column $\tilde{j}$.] Hence, if we remove players $\tilde{i}$ and $\tilde{j}$ from the game, the algorithm will run exactly as before, i.e., there will be an infinite number of consecutive pivots with $-\Delta u_i = 0 \ \forall i$. So, by the inductive hypothesis,

Claim 1: *The matrix* $C^{tR}_{-\tilde{i}\tilde{j}}$ *[By* $C^{tR}_{-\tilde{i}\tilde{j}}$ *we mean the matrix* C^{tR} *without the* $\tilde{i}$*th row and the* j*th column.] has the D-condition.*

Next, since $\mu^t(i) \neq \tilde{j}\ \forall i \neq \tilde{i}$ (see above), we have

Claim 2: $c^{tR}_{i\tilde{j}} \leq \max_j c^{tR}_{ij}\ \forall i \neq \tilde{i}$.

Finally, since $\mu^t(\tilde{i}) = \tilde{j}$, we have

Claim 3: $c^{tR}_{\tilde{i}\tilde{j}} \geq c^{tR}_{ij}\ \forall j \neq \tilde{j}$.

But Claims 1,2, and 3 together imply that C^{tR} has the D-condition, with the same $\hat{I}, \hat{J}, \sigma$ as $C^{tR}_{-\tilde{i}\tilde{j}}$, except also $\sigma(\tilde{i}) = \tilde{j}$. This concludes the proof of Theorem 4.3.

Theorem 4.3 raises some other issues to which we now turn our attention.

Conjecture: *Nondegeneracy does not necessarily imply convergence in the general Demange-Gale case.*

By nondegeneracy here, we mean that the set U (section 2) is n-dimensional.[13] At any rate, the pertinent concept in determining algorithm convergence is that of D-condition. As we have just seen, the D-condition implies nondegeneracy in the Shapley-Shubik case. However, we do not believe that (the appropriate analogue of) the D-condition implies degeneracy in the general Demange-Gale case.

The concept of D-condition also plays a vital role in terms of the rate of the algorithm's convergence. Consider the two Shapley-Shubik markets below. The matrix C_1 has the D-condition, and the algorithm does not converge. Comparing C_1 and C_2, we could say that C_2 "almost" has the D-condition. Interestingly

[13] This corresponds to the definition of nondegeneracy given in the Shapley-Shubik case (p. 23) because of Theorem 4.3 in Quint (1988).

enough, for small ϵ, the algorithm "almost" doesn't converge—in fact, it takes on the order of $2/\epsilon$ iterations to run.

$$C_1 = \begin{pmatrix} 4 & 4 & 0 \\ 4 & 4 & 1 \\ 4 & 4 & 2 \end{pmatrix}$$

$$C_2 = \begin{pmatrix} 4 & 4 & 0 \\ 4 & 4-\epsilon & 1 \\ 4 & 4 & 2 \end{pmatrix}$$

Thus, we are forced to conclude that

Remark: The algorithm is *not* polynomial in the amount of input data.

APPENDIX

Definition: A matrix C has the D-condition if $\exists$ a set of rows $\hat{I} = \{i_1, ..., i_\ell\}$ with $|\hat{I}| \geq 1$, a set of columns $\hat{J} = \{j_1, ..., j_m\}$, and a mapping $\sigma : \hat{I}^c \longrightarrow J$ with:

1) For every $i \epsilon \hat{I}$, the two maximal elements of row i are in columns of $\hat{J}$

2) For every $i \epsilon \hat{I}$, the two maximal elements of row i are equal

3) For any $i_1, i_2 \epsilon \hat{I}^c$,

A) $\sigma(i_1) \epsilon \arg\max_j c_{i_1 j}$

B) $\sigma(i_1) \notin \hat{J}$

C) $\sigma(i_1) \neq \sigma(i_2)$

Lemma: *If an $n \times n$ matrix has the D-condition, it is degenerate.*

Proof: We begin with a Proposition:

Proposition: *Suppose an $n \times n$ matrix C has the following properties ($n \geq 2$):*

1) In each row, the top two elements are equal.

2) For each column j^*, $j^* \epsilon \arg\max_j c_{ij}$ for some i.

Then C is degenerate, and, for every maximal matching μ^*, $\mu^*(i) \epsilon \arg\max_j c_{ij}$ $\forall i$.

Proof: By induction on n. If $n = 2$, both rows of C necessarily have two identical elements, so the Proposition trivially holds. So suppose its true for $n-1$-dimensional matrices.

Case 1: $\exists$ column $\tilde{j}$ with only one i, say $\tilde{i}$, satisfying $\tilde{j} \epsilon \arg\max_j c_{ij}$.

In this case, remove row $\tilde{i}$ and column $\tilde{j}$ from the game. The hypotheses now hold for the $(n-1) \times (n-1)$ matrix $C_{-\tilde{i}\tilde{j}}$. Let $\mu^{*1}_{-\tilde{i}\tilde{j}}$ and $\mu^{*2}_{-\tilde{i}\tilde{j}}$ be two maximal matchings for $C_{-\tilde{i}\tilde{j}}$. Then define

$$\mu^{*1}(i) = \mu^{*1}_{-\tilde{i}\tilde{j}}(i) \text{ for } i \neq \tilde{i},$$

$$\mu^{*1}(\tilde{i}) = \tilde{j},$$

$$\mu^{*2}(i) = \mu^{*2}_{-\tilde{i}\tilde{j}}(i) \text{ for } i \neq \tilde{i},$$

$$\mu^{*2}(\tilde{i}) = \tilde{j}.$$

Then μ^{*1} and μ^{*2} satisfy the Proposition.

Case 2: All columns j^* satisfy $|K(j^*)| = |\{i : j^* \epsilon \arg\max_j c_{ij}\}| \geq 2$.

Case 2A: $\exists$ row $\tilde{i}$ with at least 3 equal maximal elements.

In this case, let $\tilde{j} \epsilon \arg\max_j c_{\tilde{i}j}$. Then in a manner similar to Case 1, we can eliminate row $\tilde{i}$ and column $\tilde{j}$ and use the inductive hypothesis.

Case 2B: $\nexists$ such a row.

Then the matrix, after relabeling columns, has the form:

$$\hat{C} = \begin{pmatrix} x & x & & \dots & & \\ & x & x & \dots & & \\ & & x & \dots & & \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ & & & \dots & x & x \\ x & & & \dots & & x \end{pmatrix},$$

where cell $(\tilde{i}, \tilde{j})$ containing an "x" means $\tilde{j} \epsilon \arg\max_j c_{\tilde{i}j}$. But then, $\mu^{*1} : \mu^{*1}(i) = i$ and $\mu^{*2} : \mu^{*2}(i) = i + 1 \pmod{n}$ both satisfy the requirements of the Proposition.

Suppose now that C has the D-condition with $\hat{I}, \hat{J}, \sigma$. Denote by $\hat{\hat{J}}$ the set of columns $\{\hat{j} \epsilon \hat{J} : \hat{j} \epsilon \arg\max_j c_{ij}$ for some $i \epsilon \hat{I}\}$. Clearly C has the D-condition with $\hat{I}, \hat{\hat{J}}, \sigma$ also.

Now we can prove the Lemma. Let $\tilde{\mu}$ be any matching. We need to show that $\exists$ two matchings μ^{*1} and μ^{*2} with

$$\sum_{i=1}^{n} c_{i\mu^{*1}(i)} \geq \sum_{i=1}^{n} c_{i\tilde{\mu}(i)}, \quad \sum_{i=1}^{n} c_{i\mu^{*2}(i)} \geq \sum_{i=1}^{n} c_{i\tilde{\mu}(i)}, \text{ and } \sum_{i=1}^{n} c_{i\mu^{*1}(i)} = \sum_{i=1}^{n} c_{i\mu^{*2}(i)}.$$

To do this, define

$$\mu^{*1}(i) = \mu^{*2}(i) = \sigma(i) \text{ for } i \epsilon \hat{I}^c,$$

$$\mu^{*1}(i) = \mu^{*2}(i) = \tilde{\mu}(i) \text{ for } i \epsilon \hat{I} : \tilde{\mu}(i) \notin \{\hat{\hat{J}} \cup [\cup_{i \epsilon \hat{I}^c} \sigma(i)]\}.$$

This leaves $\mu^{*1}(i)$ and $\mu^{*2}(i)$ undefined for $i \epsilon \hat{I}$ with $\tilde{\mu}(i) \epsilon \{\hat{\hat{J}} \cup [\cup_{i \epsilon \hat{I}^c} \sigma(i)]\}$. Denote this set by $\tilde{I}$.

This also leaves $\mu^{*1\ -1}(j)$ and $\mu^{*2\ -1}(j)$ undefined for $j \epsilon \hat{\tilde{J}}$. So, $|\tilde{I}| = |\hat{\tilde{J}}|$.

Now consider the definition of the D-condition. Because of condition 3), necessarily $|\hat{I}| \geq |\hat{J}|$ (if C is square). So, since $|\hat{I}| \geq 1 \Longrightarrow |\hat{J}| \geq 2$, we have that $|\hat{I}|$ and $|\hat{J}| \geq 2$. Thus, $|\tilde{I}| = |\hat{\tilde{J}}| \geq 2$. Also, it is easy to see that the submatrix of C formed by the rows of $\tilde{I}$ and the columns of $\hat{\tilde{J}}$ satisfies the conditions for the Proposition. Denote this submatrix by $\tilde{C}$.

Let μ^1 and μ^2 be maximal matchings of $\tilde{C}$. Now finish the definition of μ^{*1} and μ^{*2} by setting

$$\mu^{*1}(i) = \mu^1(i) \text{ and } \mu^{*2}(i) = \mu^2(i) \text{ for } i \epsilon \tilde{I}.$$

Since $\mu^{*1}(i) = \mu^{*2}(i)$, $c_{i\mu^{*1}(i)} = c_{i\mu^{*2}(i)}$ for $i \notin \tilde{I}$. But also $c_{i\mu^{*1}(i)} = c_{i\mu^{*2}(i)}$ for $i \epsilon \tilde{I}$, because of the second conclusion of the Proposition. Hence, $\sum c_{i\mu^{*1}(i)} = \sum c_{i\mu^{*2}(i)}$.

As far as $\tilde{\mu}$ is concerned, we have

1) For $i \epsilon \hat{I}^c$, $c_{i\mu^{*1}(i)} \geq c_{i\tilde{\mu}(i)}$ and $c_{i\mu^{*2}(i)} \geq c_{i\tilde{\mu}(i)}$, because $\sigma(i)$ is an argmax of c_{ij} over j.

2) For $i \epsilon \hat{I}$ but $i \notin \tilde{I}$, $c_{i\mu^{*1}(i)} = c_{i\mu^{*2}(i)} = c_{i\tilde{\mu}(i)}$ because $\mu^{*1}(i) = \mu^{*2}(i) = \tilde{\mu}(i)$.

3) For $i \epsilon \tilde{I}$, remember that the second part of the conclusion to the Proposition implies that

$$\mu^1(i) \epsilon \arg \max_{j \epsilon \hat{\tilde{J}}} c_{ij} \text{ and } \mu^2(i) \epsilon \arg \max_{j \epsilon \hat{\tilde{J}}} c_{ij}.$$

But since $\tilde{I} \subset \hat{I}$, this implies

$$\mu^1(i) \epsilon \arg \max_{j \epsilon J} c_{ij} \text{ and } \mu^2(i) \epsilon \arg \max_{j \epsilon J} c_{ij}.$$

Thus, $c_{i\mu^{*1}(i)} \geq c_{i\tilde{\mu}(i)}$ and $c_{i\mu^{*2}(i)} \geq c_{i\tilde{\mu}(i)}$.

But then 1), 2), and 3) together imply $\sum c_{i\mu^{*1}(i)} \geq \sum c_{i\tilde{\mu}(i)}$ and $\sum c_{i\mu^{*2}(i)} \geq \sum c_{i\tilde{\mu}(i)}$. This concludes the proof of the Lemma.

REFERENCES

Crawford, Vincent, and E.M. Knoer, "Job Matching with Heterogeneous Firms and Workers", Econometrica, Volume 49, 1981, pp. 437-450.

Demange, Gabrielle, and David Gale, "The Strategy Structure of Two-Sided Matching Markets", Econometrica, Volume 53, 1985, pp. 873-888.

Gale, David, and Lloyd Shapley, "College Admissions and the Stability of Marriage", American Mathematical Monthly 69, pp. 9-15.

Kaneko, Mamoru, "The Central Assignment Game and the Assignment Markets", Journal of Mathematical Economics, 10, 1982, pp. 205-232.

Kelso, Alexander Jr., and Vincent Crawford, "Job Matching, Coalition Formation, and Gross Substitutes", Econometrica, Volume 50, November 1982, pp. 1483-1504.

Quint, Thomas, "A Proof of the Nonemptiness of the Core of Two-Sided Matching Markets", UCLA Computational and Applied Mathematics Report #87-29, Department of Mathematics, UCLA, Los Angeles, California, 1987.

Quint, Thomas, "Elongation of the Core in an Assignment Game", IMSSS Report, Encina Hall, Stanford University, Stanford, California, 1988.

Quinzii, Martine, "Core and Competitive Equilibria with Indivisibilities", International Journal of Game Theory, Volume 13, Issue 1, 1984, pp. 41-60.

Roth, Alvin, "Stability and Polarization of Interests in Job Matching", Econometrica 52, 1984, pp. 47-58.

Shapley, Lloyd, and Martin Shubik, "The Assignment Game I: The Core", International Journal of Game Theory, 1, 1972, pp. 111-130.

Veinott, Arthur F., Jr., "Lecture Notes in Lattice Programming", unpublished mimeograph, Department of Operations Research, Stanford University, Stanford, California, Winter 1987.