ELSEVIER Theoretical Computer Science 233 (2000) 297–308

Theoretical Computer Science

www.elsevier.com/locate/tcs

Note

A sublinear parallel algorithm for stable matching

Tomás Feder[a], Nimrod Megiddo[b,*], Serge A. Plotkin[c]

[a] *IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA*
[b] *IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120-6099, USA*
[c] *Department of Computer Science, Stanford University, Stanford, CA 94305, USA*

Received July 1995; accepted February 1999
Communicated by F. Yao

Abstract

A parallel algorithm for the stable matching problem is presented. The algorithm is based on the primal-dual interior path-following method for linear programming. The main result is that a stable matching can be found in $O^*(\sqrt{m})$ time by a polynomial number of processors, where m is the total length of preference lists of individuals. © 2000 Published by Elsevier Science B.V. All rights reserved.

Keywords: Non-expansive circuits; Linear programming; Stable matching

1. Introduction

In this paper we consider networks made of gates of constant size. We focus on non-expansive networks (to be defined below). The problems of evaluating the gate to which a network converges, and of finding a stable configuration in a network, are quite simple in the context of sequential computation; they can all be solved in linear time in the scatter-free case (a special case, [5]), and in quadratic time in the non-expansive case [1]. An interesting question is the existence of sublinear parallel algorithms with a polynomial number of processors.

We present parallel algorithms for the above problems which run in $O^*(\sqrt{I})$ time, with a polynomial number of processors, where I is the size of the input and $f(I) = O^*(g(I))$ means that there exists a constant k such that $f(I) \leqslant g(I)(\log I)^k$. Our approach is based on formulating the problems as linear programming problems and solving them with the primal-dual interior path-following method.

* Corresponding author.
E-mail address: megiddo@almaden.ibm.com (N. Megiddo)

PII: S0304-3975(99)00125-5

As an application, the problem of stable matching [3] can be solved in $O^*(\sqrt{m})$ time, where m is the total length of the preference lists of individuals.

In Sections 2 and 3 we introduce networks of gates and the concepts of non-expansive and convergent networks. The material in these sections is from Feder [1]. In Section 4 we study the relation between these concepts and linear programming. In Section 5 we obtain the general result of recognizing stability in a network. This result is then applied to the stable matching problem in Section 6.

2. Gates and networks

A (boolean) *assignment* is a mapping $\boldsymbol{x}: S \to \{0,1\}$ with a domain $S = S(\boldsymbol{x})$. An element $i \in S(\boldsymbol{x})$ is a *coordinate* of $\boldsymbol{x}$, and the image $\boldsymbol{x}(i)$ is its *value*. Given a set of coordinates $T \subseteq S(\boldsymbol{x})$, we denote by $\boldsymbol{x}_T$ the restriction of $\boldsymbol{x}$ to the set T. If $T = \{i\}$ $(i \in S(\boldsymbol{x}))$ then $\boldsymbol{x}_T$ is denoted by x_i. If $\boldsymbol{x}$ and $\boldsymbol{y}$ are assignments with $S(\boldsymbol{x}) \cap S(\boldsymbol{y}) = \emptyset$, then $\boldsymbol{xy}$ denotes the *union* of the two assignments, with $S(\boldsymbol{xy}) = S(\boldsymbol{x}) \cup S(\boldsymbol{y})$. In particular, if $S(\boldsymbol{x}) = \{1,2,\ldots,n\}$, then $\boldsymbol{x} = x_1 x_2 \ldots x_n$. With a slight compromise of notation, we shall identify each x_i with its value $\boldsymbol{x}(i)$. For example, the statement $\boldsymbol{x} = x_1x_2x_3 = 011$ indicates that $S(\boldsymbol{x}) = \{1,2,3\}$ and $(\boldsymbol{x}(1), \boldsymbol{x}(2), \boldsymbol{x}(3)) = (0,1,1)$. Two assignments $\boldsymbol{x}$ and $\boldsymbol{y}$ are *consistent* if $x_i = y_i$ for all $i \in S(\boldsymbol{x}) \cap S(\boldsymbol{y})$.

A *gate* is a mapping $g: \{0,1\}^I \to \{0,1\}^O$ from assignments on the input set $I = I(g)$ to assignments on the output set $O = O(g)$. The coordinates in $I(g)$ and $O(g)$ are called *inputs* and *outputs* of g, respectively. The gate g is a k-input, ℓ-output gate if $|I(g)| = k$ and $|O(g)| = \ell$. Given a gate g, an assignment $\boldsymbol{x}$ with $S(\boldsymbol{x}) \subseteq I(g)$ and a coordinate set $T \subseteq O(g)$, the *restriction* $g_{\boldsymbol{x},T}$ of the gate g is the gate g' obtained from g by discarding the outputs not in T and discarding the inputs in $S(\boldsymbol{x})$ after assigning to them the values given by $\boldsymbol{x}$. More formally, the gate g' has inputs $I(g') = I(g) \backslash S(\boldsymbol{x})$, outputs $O(g') = O(g) \cap T$, and satisfies $g'(\boldsymbol{y}) = g(\boldsymbol{xy})_T$.

A *network* is a set of gates that share neither inputs nor outputs. This means that if N is a network and g and g' are distinct gates in N, then $I(g) \cap I(g') = \emptyset$ and $O(g) \cap O(g') = \emptyset$. On the other hand, given two (not necessarily distinct) gates g, g' in N, it may happen that an output of g is also an input of g'. If $i \in O(g) \cap I(g')$, then we say that output i of gate g and input i of gate g' are *linked*. By the disjointness property, every input is linked to at most one output, and every output is linked to at most one input. These links induce a topology on the network that can be described by a directed multigraph on the gates of the network, i.e., a directed graph with the gates as the vertices, with a directed edge from g to g' for every output of g linked to an input of g'; loops and parallel edges are allowed. If the underlying directed multigraph of a network is acyclic, the network is called a *circuit*.

The *transition function* of a network N is a single gate f which is equivalent to the entire network as we explain below. The gate f has $I(f) = \bigcup_{g \in N} I(g)$ and $O(f) = \bigcup_{g \in N} O(g)$, and satisfies $\boldsymbol{y} = f(\boldsymbol{x})$ if and only if $\boldsymbol{y}_{O(g)} = g(\boldsymbol{x}_{I(g)})$ for all gates $g \in N$. Note that if f is the transition function of N, then the networks N and $N' = \{f\}$

have the same transition function; we shall see that, for many purposes, they can actually be treated as the same network. The set $R(N)=I(f)\cup O(f)$ of a network N with transition function f is the set of *coordinates* of the network N. It consists of three disjoint subsets: the set of *links* $L(N)=I(f)\cap O(f)$, the set of *inputs* $I(N)=I(f)\backslash O(f)$, and the set of *outputs* $O(N)=O(f)\backslash I(f)$ of the network.

A *configuration* of a network N is an assignment $\boldsymbol{u}$ on the coordinate set $R(N)$, and consists of an *input assignment* $\boldsymbol{u}_{I(N)}$, an *output assignment* $\boldsymbol{u}_{O(N)}$, and an *internal assignment* $\boldsymbol{u}_{L(N)}$. A network N can be used to define an associated mapping on the configurations of N. Given two configurations $\boldsymbol{x}$ and $\boldsymbol{y}$ of a network N with transition function f, we write $\boldsymbol{y}=N(\boldsymbol{x})$ if $\boldsymbol{y}_{I(N)}=\boldsymbol{x}_{I(N)}$ and $\boldsymbol{y}_{O(N)\cup L(N)}=f(\boldsymbol{x}_{I(N)\cup L(N)})$. In other words, all gates are evaluated using the values assigned to their inputs by $\boldsymbol{x}$, thus obtaining at their outputs the values for the configuration $\boldsymbol{y}$; the inputs to the network are not outputs of any gate, and thus keep their value from $\boldsymbol{x}$. A configuration $\boldsymbol{x}$ is *stable* if $N(\boldsymbol{x})=\boldsymbol{x}$. Thus, a configuration $\boldsymbol{x}$ is stable if it satisfies $f(\boldsymbol{x}_{I(f)})=\boldsymbol{x}_{O(f)}$ for the transition function f or, equivalently, $g(\boldsymbol{x}_{I(g)})=\boldsymbol{x}_{O(g)}$ for each gate $g\in N$, i.e., if it satisfies all the gate equations.

The kth *iterate* of a mapping τ on a set U is the mapping $\tau^{(k)}$ defined inductively by letting $\tau^{(0)}(z)=z$ and $\tau^{(k+1)}(z)=\tau(\tau^{(k)}(z))$ for all $z\in U$. A *periodic point* of τ is a z such that $\tau^{(p)}(z)=z$ for some $p\geqslant 1$. The least such p is the *period* of z. A *fixed point* of τ is a periodic point of period 1. We are particularly interested in the iterates and periodic points of the mapping associated with a network N. It will sometimes be useful to look at the iterates $N^{(k)}$ in terms of the transition function f of the network. For this purpose, we define two restrictions of f given an input assignment for the network. Given an assignment $\boldsymbol{x}$ on $I(N)$, the *output function* of the network is the mapping $g_{\boldsymbol{x}}=f_{\boldsymbol{x},O(N)}$, and the *internal function* of the network is the mapping $h_{\boldsymbol{x}}=f_{\boldsymbol{x},L(N)}$, so that if $\boldsymbol{z}$ is an assignment on $L(N)$, then $f(\boldsymbol{xz})=g_{\boldsymbol{x}}(\boldsymbol{z})h_{\boldsymbol{x}}(\boldsymbol{z})$. If $\boldsymbol{y}$ is an assignment on $O(N)$, then $N(\boldsymbol{xyz})=\boldsymbol{x}g_{\boldsymbol{x}}(\boldsymbol{z})h_{\boldsymbol{x}}(\boldsymbol{z})$, and $N^{(k+1)}(\boldsymbol{xyz})=\boldsymbol{x}g_{\boldsymbol{x}}(h_{\boldsymbol{x}}^{(k)}(\boldsymbol{z}))h_{\boldsymbol{x}}^{(k+1)}(\boldsymbol{z})$ for all $k\geqslant 0$. The periodic points of the mapping associated with N are called *periodic configurations*; the fixed points are precisely the stable configurations. The periodic configurations $\boldsymbol{xyz}$ consistent with an input assignment $\boldsymbol{x}$ are determined by the choice of a periodic point $\boldsymbol{z}$ of the internal function $h_{\boldsymbol{x}}$. For if $\boldsymbol{z}$ has period p and $\boldsymbol{z}'=h_{\boldsymbol{x}}^{(p-1)}(\boldsymbol{z})$, then the periodic configuration must have $\boldsymbol{z}=h_{\boldsymbol{x}}(\boldsymbol{z}')$ and $\boldsymbol{y}=g_{\boldsymbol{x}}(\boldsymbol{z}')$. Thus, the periodic configurations are the configurations $\boldsymbol{x}g_{\boldsymbol{x}}(\boldsymbol{z}')h_{\boldsymbol{x}}(\boldsymbol{z}')$ with $\boldsymbol{z}'$ a periodic point of $h_{\boldsymbol{x}}$.

3. Non-expansive mappings and convergent networks

The *distance* $d(\boldsymbol{x},\boldsymbol{y})$ between two assignments $\boldsymbol{x}$ and $\boldsymbol{y}$ on a set S is defined by

$$d(\boldsymbol{x},\boldsymbol{y})=\sum_{i\in S}|x_i-y_i|.$$

A gate g is *non-expansive* if for any two assignments $\boldsymbol{x}$ and $\boldsymbol{y}$ on $I(g)$,

$$d(g(\boldsymbol{x}),g(\boldsymbol{y}))\leqslant d(\boldsymbol{x},\boldsymbol{y}).$$

A network N is said to be *convergent* if for every input assignment $\boldsymbol{x}$ there exists an output assignment $\boldsymbol{y}$ such that every configuration consistent with $\boldsymbol{x}$ maps to a configuration consistent with $\boldsymbol{y}$ under sufficiently many iterations of N. More precisely, for every configuration $\boldsymbol{u}$ consistent with $\boldsymbol{x}$, there exists an integer k_0 such that $N^{(k)}(\boldsymbol{u})$ is consistent with $\boldsymbol{y}$ for all $k \geqslant k_0$. Since every configuration maps to a periodic configuration for sufficiently large k, and every periodic configuration maps to itself for infinitely many values of k, the condition of convergence is equivalent to the requirement that every periodic configuration consistent with $\boldsymbol{x}$ must also be consistent with $\boldsymbol{y}$. Recall that the periodic configurations of N are the configurations $\boldsymbol{x}g_{\boldsymbol{x}}(\boldsymbol{z})h_{\boldsymbol{x}}(\boldsymbol{z})$, where $\boldsymbol{z}$ is a periodic point of $h_{\boldsymbol{x}}$ and the mappings $g_{\boldsymbol{x}}, h_{\boldsymbol{x}}$ are the output and internal functions of N (see Section 2). The condition defining convergent networks becomes then the statement that $g_{\boldsymbol{x}}(\boldsymbol{z}) = \boldsymbol{y}$ for all periodic points $\boldsymbol{z}$ of $h_{\boldsymbol{x}}$. If a network N is convergent, then for every input assignment $\boldsymbol{x}$ there is a *unique* corresponding output assignment $\boldsymbol{y}$ for N, and we say that N *converges* to the gate g with $I(g) = I(N)$ and $O(g) = O(N)$ that computes $g(\boldsymbol{x}) = \boldsymbol{y}$.

The notion of a convergent network evolved out of discussions between the first author and Ashok Subramanian, and was motivated by the following observation.

Lemma 3.1. *Every network of non-expansive gates converges to a non-expansive gate.*

Proof. Let f be the transition function of a network N of non-expansive gates, given by $f(\boldsymbol{x}\boldsymbol{z}) = g_{\boldsymbol{x}}(\boldsymbol{z})h_{\boldsymbol{x}}(\boldsymbol{z})$, where $g_{\boldsymbol{x}}$ and $h_{\boldsymbol{x}}$ are the output and internal functions. Let $\boldsymbol{z}$ and $\boldsymbol{z}'$ be periodic points of $h_{\boldsymbol{x}}$. Under these conditions,

$$\begin{aligned} d(g_{\boldsymbol{x}}(\boldsymbol{z}), g_{\boldsymbol{x}}(\boldsymbol{z}')) + d(h_{\boldsymbol{x}}(\boldsymbol{z}), h_{\boldsymbol{x}}(\boldsymbol{z}')) &= d(f(\boldsymbol{x}\boldsymbol{z}), f(\boldsymbol{x}\boldsymbol{z}')) \\ &\leqslant d(\boldsymbol{x}\boldsymbol{z}, \boldsymbol{x}\boldsymbol{z}') = d(\boldsymbol{z}, \boldsymbol{z}') = d(h_{\boldsymbol{x}}(\boldsymbol{z}), h_{\boldsymbol{x}}(\boldsymbol{z}')), \end{aligned}$$

so $d(g_{\boldsymbol{x}}(\boldsymbol{z}), g_{\boldsymbol{x}}(\boldsymbol{z}')) = 0$ and $g_{\boldsymbol{x}}(\boldsymbol{z}) = g_{\boldsymbol{x}}(\boldsymbol{z}')$. Therefore, the output $\boldsymbol{y} = g_{\boldsymbol{x}}(\boldsymbol{z})$ depends only on $\boldsymbol{x}$, and not on the choice of a periodic point $\boldsymbol{z}$. This shows that the network is convergent, and converges to some gate g, where $g(\boldsymbol{x}) = g_{\boldsymbol{x}}(\boldsymbol{z})$ for all periodic points $\boldsymbol{z}$ of $h_{\boldsymbol{x}}$.

Given two input assignments $\boldsymbol{x}$ and $\boldsymbol{x}'$, let $\boldsymbol{z}$ and $\boldsymbol{z}'$ be periodic points of $h_{\boldsymbol{x}}$ and $h_{\boldsymbol{x}'}$, respectively, that are closest to each other. In particular, $d(\boldsymbol{z}, \boldsymbol{z}') = d(h_{\boldsymbol{x}}(\boldsymbol{z}), h_{\boldsymbol{x}'}(\boldsymbol{z}'))$. Then

$$\begin{aligned} d(g_{\boldsymbol{x}}(\boldsymbol{z}), g_{\boldsymbol{x}'}(\boldsymbol{z}')) + d(h_{\boldsymbol{x}}(\boldsymbol{z}), h_{\boldsymbol{x}'}(\boldsymbol{z}')) &= d(f(\boldsymbol{x}\boldsymbol{z}), f(\boldsymbol{x}'\boldsymbol{z}')) \\ &\leqslant d(\boldsymbol{x}\boldsymbol{z}, \boldsymbol{x}'\boldsymbol{z}') = d(\boldsymbol{x}, \boldsymbol{x}') + d(\boldsymbol{z}, \boldsymbol{z}') \\ &\leqslant d(\boldsymbol{x}, \boldsymbol{x}') + d(h_{\boldsymbol{x}}(\boldsymbol{z}), h_{\boldsymbol{x}'}(\boldsymbol{z}')). \end{aligned}$$

Thus, $d(g(\boldsymbol{x}), g(\boldsymbol{x}')) = d(g_{\boldsymbol{x}}(\boldsymbol{z}), g_{\boldsymbol{x}'}(\boldsymbol{z}')) \leqslant d(\boldsymbol{x}, \boldsymbol{x}')$, so the gate g is non-expansive. $\square$

4. Convergent networks and linear programming

4.1. Linear characterizations of stability

Let N be a non-expansive network with transition function f. A configuration $\boldsymbol{x}$ is stable if and only if for every configuration $\boldsymbol{a}$,

$$d(f(\boldsymbol{a}_{I(f)}), \boldsymbol{x}_{O(f)}) \leqslant d(\boldsymbol{a}_{I(f)}, \boldsymbol{x}_{I(f)}). \tag{1}$$

For every fixed $\boldsymbol{a}$, this is a linear inequality in terms of $\boldsymbol{x}$ since, for example,

$$d(\boldsymbol{a}, \boldsymbol{x}) = \sum_{i:a_i=0} x_i + \sum_{i:a_i=1} (1 - x_i).$$

Definition 4.1. Denote by $\mathscr{A}$ the system of all the linear inequalities (1) corresponding to the configurations $\boldsymbol{a}$, the inequalities $0 \leqslant x_i \leqslant 1$, and the initial assignments $\boldsymbol{x}_{I(N)} = \boldsymbol{a}_{I(N)}$ for some fixed input assignment $\boldsymbol{a}_{I(N)}$. Denote by n the number of variables in $\mathscr{A}$.

Proposition 4.2. *The system $\mathscr{A}$ has a solution.*

Proof. Let us extend the mapping f into a continuous multilinear mapping $\tilde{f}$ on the full hypercube $[0, 1]^n$, by defining

$$\tilde{f}(\boldsymbol{x}) = \sum_{\boldsymbol{a}} f(\boldsymbol{a}) \prod_{i \in I(f)} w(\boldsymbol{x}, \boldsymbol{a}, i),$$

where the summation ranges over all $\{0, 1\}$-configurations $\boldsymbol{a}$, and

$$w(\boldsymbol{x}, \boldsymbol{a}, i) = \begin{cases} x_i & \text{if } a_i = 1 \\ 1 - x_i & \text{otherwise.} \end{cases}$$

We claim that the extension $\tilde{f}$ is also non-expansive, i.e., for any two vectors $\boldsymbol{x}, \boldsymbol{y} \in [0, 1]^n$, $d(\tilde{f}(\boldsymbol{x}), \tilde{f}(\boldsymbol{y})) \leqslant d(\boldsymbol{x}, \boldsymbol{y})$. The proof is as follows. Given any two points $\boldsymbol{x}, \boldsymbol{y} \in [0, 1]^n$, consider the points $\boldsymbol{p}^j = (y_1, \dots, y_{j-1}, x_j, \dots, x_n)$, $j = 1, \dots, n+1$. Since $d(\boldsymbol{p}^j, \boldsymbol{p}^{j+1}) = |x_j - y_j|$, it follows that

$$d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{p}^1, \boldsymbol{p}^2) + d(\boldsymbol{p}^2, \boldsymbol{p}^3) + \cdots + d(\boldsymbol{p}^n, \boldsymbol{p}^{n+1}).$$

Since $\boldsymbol{p}^1 = \boldsymbol{x}$ and $\boldsymbol{p}^{n+1} = \boldsymbol{y}$, by the triangular inequality,

$$d(\tilde{f}(\boldsymbol{x}), \tilde{f}(\boldsymbol{y})) \leqslant d(\tilde{f}(\boldsymbol{p}^1), \tilde{f}(\boldsymbol{p}^2)) + \cdots + d(\tilde{f}(\boldsymbol{p}^n), \tilde{f}(\boldsymbol{p}^{n+1})).$$

Thus, it suffices to show that for every j,

$$d(\tilde{f}(\boldsymbol{p}^j), \tilde{f}(\boldsymbol{p}^{j+1})) \leqslant d(\boldsymbol{p}^j, \boldsymbol{p}^{j+1}).$$

For $j \geqslant 1$, denote

$$\boldsymbol{q}^j = (y_1, \dots, y_{j-1}, 0, x_{j+1}, \dots, x_n),$$

and

$$\boldsymbol{r}^j = (y_1, \dots, y_{j-1}, 1, x_{j+1}, \dots, x_n)$$

and denote by $\boldsymbol{e}^j$ a unit vector with a 1 in the jth position. By definition,

$$\tilde{f}(\boldsymbol{p}^j) = x_j \tilde{f}(\boldsymbol{r}^j) + (1 - x_j)\tilde{f}(\boldsymbol{q}^j)$$

and

$$\tilde{f}(\boldsymbol{p}^{j+1}) = y_j \tilde{f}(\boldsymbol{r}^j) + (1 - y_j)\tilde{f}(\boldsymbol{q}^j),$$

so

$$d(\tilde{f}(\boldsymbol{p}^{j+1}), \tilde{f}(\boldsymbol{p}^j)) = \|(y_j - x_j)(\tilde{f}(\boldsymbol{r}^j) - \tilde{f}(\boldsymbol{q}^j))\|_1 = |y_j - x_j| \cdot d(\tilde{f}(\boldsymbol{r}^j), \tilde{f}(\boldsymbol{q}^j)).$$

Now, for every $i \neq j$, and every $\boldsymbol{a}$ such that $a_j = 0$, $w(\boldsymbol{r}^j, \boldsymbol{a} + e^j, i) = w(\boldsymbol{q}^j, \boldsymbol{a}, i)$, so

$$\begin{aligned}\tilde{f}(\boldsymbol{r}^j) - \tilde{f}(\boldsymbol{q}^j) &= \sum_{\boldsymbol{a}:a_j=1} f(\boldsymbol{a}) \prod_{i\in I(f)\setminus\{j\}} w(\boldsymbol{r}^j, \boldsymbol{a}, i) - \sum_{\boldsymbol{a}:a_j=0} f(\boldsymbol{a}) \prod_{i\in I(f)\setminus\{j\}} w(\boldsymbol{q}^j, \boldsymbol{a}, i)\\ &= \sum_{\boldsymbol{a}:a_j=0} (f(\boldsymbol{a} + e^j) - f(\boldsymbol{a})) \prod_{i\in I(f)\setminus\{j\}} w(\boldsymbol{r}^j, \boldsymbol{a}, i).\end{aligned}$$

Thus,

$$\begin{aligned}d(\tilde{f}(\boldsymbol{r}^j), \tilde{f}(\boldsymbol{q}^j)) &\leqslant \sum_{\boldsymbol{a}:a_j=0} d(f(\boldsymbol{a} + e^j), f(\boldsymbol{a})) \prod_{i\in I(f)\setminus\{j\}} w(\boldsymbol{r}^j, \boldsymbol{a}, i)\\ &\leqslant \max_{\boldsymbol{a}:a_j=0} d(f(\boldsymbol{a} + e^j), f(\boldsymbol{a})) \leqslant 1\end{aligned}$$

and we get

$$d(\tilde{f}(\boldsymbol{p}^{j+1}), \tilde{f}(\boldsymbol{p}^j)) \leqslant |y_j - x_j| = d(\boldsymbol{p}^j, \boldsymbol{p}^{j+1}).$$

For two assignments $\boldsymbol{y}$ and $\boldsymbol{z}$ on $L(N)$, we say that $g(\boldsymbol{y}) = \boldsymbol{z}$ if $\tilde{f}(\boldsymbol{a}_{I(N)}\boldsymbol{y}) = \boldsymbol{zu}$ for some assignment $\boldsymbol{u}$ on $O(N)$. By Brouwer's theorem, g has a fixed point $\boldsymbol{y}$, i.e., $g(\boldsymbol{y}) = \boldsymbol{y}$. Such a point $\boldsymbol{a}_{I(N)}\boldsymbol{yu}$ satisfies the conditions in $\mathscr{A}$. □

Proposition 4.3. *Given an input $\{0,1\}$-assignment $\boldsymbol{a}_{I(N)}$, the system $\mathscr{A}$ has a unique solution for those variables corresponding to the coordinates in $O(N)$. This unique solution coincides with the value of the gate to which the network converges.*

Proof. Let $\boldsymbol{x}$ be a solution of $\mathscr{A}$ as proven in Proposition 4.2. For every $\boldsymbol{a}$ and $\boldsymbol{z}$, represent $f(\boldsymbol{az}) = g_{\boldsymbol{a}}(\boldsymbol{z})h_{\boldsymbol{a}}(\boldsymbol{z})$ where $g_{\boldsymbol{a}}$ and $h_{\boldsymbol{a}}$ are the output and the internal functions, respectively. Let $\boldsymbol{z}$ be an integer periodic point of $h_{\boldsymbol{a}}$ which is closest to $\boldsymbol{x}_{L(N)}$ (recall that $\boldsymbol{x}_{L(N)}$ consists of those variables in $\boldsymbol{x}$ associated with the links of N). We have

$$d(h_{\boldsymbol{a}}(\boldsymbol{z}, \boldsymbol{x}_{L(N)}) \geqslant d(\boldsymbol{z}, \boldsymbol{x}_{L(N)}).$$

Also, from the inequalities in the system $\mathscr{A}$,

$$d(\boldsymbol{z},\boldsymbol{x}_{L(N)}) \geqslant d(f(\boldsymbol{az}),\boldsymbol{x}_{L(N)\cup O(N)}) \leqslant d(\boldsymbol{z},\boldsymbol{x}_{L(N)})$$

$$= d(h_{\boldsymbol{a}}(\boldsymbol{z}),\boldsymbol{x}_{L(N)}) + d(g_{\boldsymbol{a}}(\boldsymbol{z}),\boldsymbol{x}_{O(N)}).$$

It follows that $d(g_{\boldsymbol{a}}(\boldsymbol{z}),\boldsymbol{x}_{O(N)}) = 0$. □

Unfortunately, the size of $\mathscr{A}$ is *exponential* since there are 2^n choices for $\boldsymbol{a}$. However, we may consider the gates $g \in N$ separately, and require instead that

$$d(g(\boldsymbol{b}),\boldsymbol{x}_{O(g)}) \leqslant d(\boldsymbol{b},\boldsymbol{x}_{I(g)}), \tag{2}$$

where $\boldsymbol{b}$ ranges over the possible input assignments for each gate g. When the gates are of constant size, this gives a number of constraints that is *linear* in the number of gates.

Definition 4.4. Denote the system of linear inequalities (2), $0 \leqslant x_i \leqslant 1$, and $\boldsymbol{x}_{I(N)} = \boldsymbol{a}_{I(N)}$ by $\mathscr{B}$. Denote by the total number of variables and inequalities in $\mathscr{B}$ by m.

4.2. The primal-dual path following method

Consider a linear program of the form

$$\text{(P)}\quad \begin{array}{ll} \text{Minimize} & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \\ & \boldsymbol{x} \geqslant \boldsymbol{0}. \end{array}$$

The dual of (P) is

$$\text{(D)}\quad \begin{array}{ll} \text{Maximize} & \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \\ \text{subject to} & \boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{c}, \\ & \boldsymbol{s} \geqslant \boldsymbol{0}. \end{array}$$

The *central path* of this primal-dual pair (P, D) (Megiddo [6]) consists of all the points $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{s})$ that satisfy the constraints of (P) and (D) together with the equations

$$x_i s_i = \mu \quad (i = 1,2,\ldots),$$

where μ varies over the non-negative reals. The *duality gap* associated with $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{s})$ is given by

$$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} = \boldsymbol{s}^{\mathrm{T}}\boldsymbol{x}.$$

Kojima et al. [4] and Monteiro and Adler [7] developed polynomial-time algorithms for tracing the primal-dual central path. They showed, in particular, that for any constant

$\delta > 0$, given an initial triple $(\boldsymbol{x}^0, \boldsymbol{y}^0, \boldsymbol{s}^0)$ on the central path, the duality gap $\boldsymbol{s}^{\mathrm{T}}\boldsymbol{x}$ can be reduced in $\mathrm{O}(\sqrt{m}\,\log((\boldsymbol{s}^0)^{\mathrm{T}}\boldsymbol{x}^0))$ (where m is the total number of primal and dual variables) iterations to at most δ.

4.3. Special form linear programs

Definition 4.5. For the sake of this definition, we say that x_i and $\bar{x}_i = 1 - x_i$ are *positive* terms while $-x_i$ and $-\bar{x}_i$ are *negative* ones. A linear programming problem is said to be in *special form* if it calls for minimizing a sum ϕ of distinct positive terms, subject to a set constraints of the form $\psi \geqslant 0$, where each ψ is a sum of positive terms as well as terms of the form $-x_i$ or $-\bar{x}_i$, which we call *negative* terms. Included are the constraints $x_i \geqslant 0$ and $\bar{x}_i \geqslant 0$. Furthermore, a problem in special form has an optimal solution and the optimal value is equal to zero. The size of the problem is total number m of constraints.

Theorem 4.6. *A linear program in special form of size m can be reduced into an equivalent form such that for any constant $\delta > 0$, after $\mathrm{O}(\sqrt{m}\,\log m)$ iterations, the duality gap of the current solution is at most δ.*

Proof. The idea of the proof is to transform the problem that is given in special form into an equivalent linear programming problem for which a point on the central path is readily available, and the size of which is not greater than some constant times the size of the given problem. At that point on the central path, each primal variable is equal to $\frac{1}{2}$ and each dual slack variable is equal to 1. Starting the algorithm of [4] or [7] from such a point, it takes $\mathrm{O}(\sqrt{m}\,\log m)$ iterations to get to a point where the duality gap is at most δ.

The transformation is carried out as follows. Let $\psi \geqslant 0$ be any constraint of the given problem, other than a non-negativity constraint $x_i \geqslant 0$. Let $v(\psi)$ denote the value of ψ which results when all the variables in ψ are set to $\frac{1}{2}$. We distinguish cases: (i) $v(\psi) > 0$. In this case we introduce a new variable $u_\psi \geqslant 0$ and replace the constraint $\psi \geqslant 0$ by the constraint $\psi - 2v(\psi) \cdot u_\psi = 0$. (ii) $v(\psi) \leqslant 0$. In this case we introduce two new variables $u_\psi, v_\psi \geqslant 0$ and replace the constraint $\psi \geqslant 0$ by $\psi - u_\psi - (2v(\psi) - 1) \cdot v_\psi = 0$. Note that the constraints are satisfied when all the variables, including u_ψ and v_ψ are set to $\frac{1}{2}$. In case (i) the transformation produces an equivalent problem. In case (ii) the new problem would be equivalent to the given one if it were guaranteed that at an optimal solution $v_\psi = 0$. To that end, we replace the objective function ϕ by $\phi + v_\psi$. Since the original objective is a sum of non-negative terms and has a feasible solution where all of them vanish, it follows that at an optimal solution of the new problem the variables v_ψ must vanish. Thus, we have transformed the problem into a problem of the form of (P) where $c_j \in \{-1, 0, 1\}$ for every j, and the point $\boldsymbol{x}^0 = (\frac{1}{2}, \ldots, \frac{1}{2})^{\mathrm{T}}$ is a feasible solution. Also, for every feasible solution $\boldsymbol{x}$, $x_j \leqslant 1$ for every j. We now show how to transform the problem so that an initial point $(\boldsymbol{x}^0, \boldsymbol{y}^0, \boldsymbol{s}^0)$ on the central path is available. Let $\boldsymbol{s}^0 = (1, \ldots, 1)^{\mathrm{T}}$ and $\boldsymbol{y}^0 = (0, \ldots, 0)^{\mathrm{T}}$. If $c_j = 1$, then the dual constraint

$(y^{\mathrm{T}}A)_j + s_j = c_j$ is satisfied by $\boldsymbol{y}^0$ and $\boldsymbol{s}^0$. If $c_j = 0$, we add a redundant constraint $x_j + u_j = 1$, where u_j is a new non-negative slack variable) with a corresponding dual variable z_j. The initial value is $u_j^0 = \frac{1}{2}$. By choosing the initial value to be $z_j^0 = -1$, we satisfy the dual constraint $z_j^0 + ((y^0)^{\mathrm{T}}A)_j + s_j^0 = c_j$. The dual constraint corresponding to u_j is $z_j + v_j = 0$ (where v_j is a new non-negative dual slack variable). By choosing the initial value to be $v_j^0 = 1$ the latter constraint is also satisfied. Finally, if $c_j = -1$, we add a redundant constraint $2x_j + u_j + w_j = 2$, where u_j and w_j are new non-negative slack variables, with a corresponding dual variable z_j. We choose $u_j^0 = v_j^0 = \frac{1}{2}$. Again, by choosing $z_j^0 = -1$, we satisfy the dual constraint $2z_j^0 + ((y^0)^{\mathrm{T}}A)_j + s_j^0 = c_j$. The dual constraints corresponding to u_j and w_j are $z_j + v_j = z_j + r_j = 0$ (where v_j and r_j are new non-negative dual slack variables) and we choose $v_j^0 = r_j^0 = 1$. In summary, all the primal variables are initially set to $\frac{1}{2}$ and all the dual slack variables are set to 1, so the initial point is on the central path. ☐

4.4. *Convergent network evaluation*

The linear programming problem associated with a non-expansive network N is

$$\begin{aligned} \text{Minimize} \quad & d(\boldsymbol{a}_{I(N)}, \boldsymbol{x}_{I(N)}), \\ \text{subject to} \quad & d(g(\boldsymbol{b}), \boldsymbol{x}_{O(g)}) - d(\boldsymbol{b}, \boldsymbol{x}_{I(g)}) \leqslant 0, \\ & 0 \leqslant x_i \leqslant 1, \end{aligned}$$

where $\boldsymbol{a}_{I(N)}$ is the input assignment to the network, g ranges over gates and $\boldsymbol{b}$ ranges over input assignments. The linear program is thus of the special form as in Definition 4.5, so Theorem 4.6 can be applied.

The duality gap can be reduced below any constant $\delta > 0$. Let $\boldsymbol{a}$ be a periodic configuration consistent with the input assignment $\boldsymbol{a}_{I(N)}$ which is closest to $\boldsymbol{x}$. Thus,

$$d(f(\boldsymbol{a}_{I(f)})_{L(N)}, \boldsymbol{x}_{L(N)}) \geqslant d(\boldsymbol{a}_{L(N)}, \boldsymbol{x}_{L(N)}).$$

On the other hand, when the linear program is put into the equivalent form, we obtain

$$d(f(\boldsymbol{a}_{I(f)}), \boldsymbol{x}_{O(f)}) - \sum y_i \leqslant d(\boldsymbol{a}_{I(f)}, \boldsymbol{x}_{I(f)}),$$

where the y_i's are the artificial variables that were added to the objective function. This can be rewritten as

$$d(\boldsymbol{a}_{O(N)}, \boldsymbol{x}_{O(N)}) + d(f(\boldsymbol{a}_{I(f)})_{L(N)}, \boldsymbol{x}_{L(N)}) \leqslant d(\boldsymbol{a}_{L(N)}, \boldsymbol{x}_{L(N)}) + d(\boldsymbol{a}_{I(N)}, \boldsymbol{x}_{I(N)}) + \sum y_i$$

because $f(\boldsymbol{a}_{I(f)})_{O(N)} = \boldsymbol{a}_{O(N)}$. On the other hand,

$$d(\boldsymbol{a}_{I(N)}, \boldsymbol{x}_{I(N)}) + \sum y_i \leqslant \delta,$$

since all the y_i's appear in the objective function, so by combining the three inequalities we get

$$d(\boldsymbol{a}_{O(N)}, \boldsymbol{x}_{O(N)}) \leqslant \delta.$$

Provided that $\delta < \frac{1}{2}$, this can be used to obtain the value of $\boldsymbol{a}_{O(N)}$, which is the output value produced by the gate to which the non-expansive network converges.

It is observed in Goldberg et al. [2] that one iteration of the interior point algorithm can be performed in $O(\log^2 m)$ time in the concurrent-read concurrent-write (CRCW) PRAM model with m^3 processors. Thus, we have the following:

Theorem 4.7. *For any fixed integer k, and for any non-expansive network N with gates of at most k inputs and outputs, the gate to which N converges can be evaluated in* $O(\sqrt{m}\log m)$ *iterations, and in an overall parallel time of* $O(\sqrt{m}\log^3 m)$ *on an* m^3*-processor CRCW PRAM, where m is the the total number of gates in N.*

5. Convergent networks and stable configurations

Let f be a non-expansive gate with $I(f)=O(f)=T$. A *fixed point* of f is an assignment $\boldsymbol{a}$ on T such that $f(\boldsymbol{a})=\boldsymbol{a}$.

For a subset $S\subseteq T$, we define the *projection* f_S to be the gate g with $I(g)=O(g)=S$ such that $g(\boldsymbol{a})=\boldsymbol{b}$ if and only if for every periodic point, $\boldsymbol{z}$, of $f_{\boldsymbol{a},T\setminus S}$, there exists a $\boldsymbol{z}'$ such that $f(\boldsymbol{az})=\boldsymbol{bz}'$. Thus, if f is the transition function of a non-expansive network N with $I(f)=O(f)=T$, and if $S\subseteq T$, then f_S can be defined as the projection on S of the gate to which the network N converges. The following is from Feder [1].

Lemma 5.1. (i) *A non-expansive mapping f has a fixed point if and only if for every $S\subseteq T$ with $|S|=1$, f_S has a fixed point.*

(ii) *A configuration $\boldsymbol{a}$ is a fixed point of f if and only if for all $S\subseteq T$ with $|S|=2$, $\boldsymbol{a}_S$ is a fixed point of f_S.*

(iii) *In* (ii), *if f is the transition function of a network N, then S may be restricted to sets of two elements that are inputs or outputs of the same gate.*

Corollary 5.2. *The set of fixed points of f can be characterized as an instance of the 2-SATISFIABILITY problem with clauses $(x_i\neq a_i)\vee(x_j\neq a_j)$ for all i,j,a_i, and a_j such that a_ia_j is not a fixed point of $f_{\{i,j\}}$. If f is the transition function of a network N, then $\{i,j\}$ may be restricted to two elements that are inputs or outputs of the same gate.*

It follows that the question of deciding whether a non-expansive network has a stable configuration reduces to $2m$ evaluations of gates to which non-expansive networks converge, and the search for a stable configuration reduces to $4\binom{m}{2}$ evaluations of such gates; in fact only $O(m)$ evaluations are needed here if for some fixed integer k the gates have at most k inputs and outputs. Since 2-SATISFIABILITY is in the class NC, we obtain:

Theorem 5.3. *For any fixed integer k, there exists an m^4-processor $O(\sqrt{m}\log^3 m)$ time CRCW PRAM algorithm that finds a stable configuration in a non-expansive network with gates of at most k inputs and outputs.*

6. Network stability and stable matching

Definition 6.1. The *X-gate* is a 2-input, 2-output gate which on inputs x_1, x_2 produces outputs y_1, y_2, such that

$$(y_1, y_2) = \begin{cases} (0,0) & \text{if } (x_1, x_2) = (1,1) \\ (x_1, x_2) & \text{otherwise.} \end{cases}$$

It is easy to see that the X-gate is non-expansive. Subramanian [8] showed that the stable matching problem can be viewed as the problem of finding a stable configuration of a network of X-gates. The coordinates of the network are pairs ij, where i is the name of an individual and $0 \leqslant j \leqslant \ell_i$, where ℓ_i is the length of the preference list of individual i. If the jth choice of individual i is individual i', and the j'th choice of individual i' is individual i, then the network has an X-gate with the coordinates $i(j-1)$ and $i'(j'-1)$ as inputs, and the coordinates ij and $i'j'$ as outputs. The input $i0$ has the value $x_{i0} = 1$. Thus, in a stable configuration, the values x_{ij} for a fixed individual i are monotonically non-increasing. If there is an index j such that $x_{i(j-1)} = 1$ and $x_{ij} = 0$, then i is matched to his jth choice. There can be at most one such index. The outputs $i\ell_i$ indicate whether i is matched to some partner, and are independent of the choice of stable matching.

It is of interest to see what the inequalities in (2) correspond to in the case of X-gates. Recall that

$$X(00) = X(11) = 00, \quad X(01) = 01, \quad X(10) = 10.$$

For $\boldsymbol{b} = 10$, $g(\boldsymbol{b}) = X(\boldsymbol{b}) = 10$ and the non-expansiveness condition is

$$(1 - y_1) + y_2 \leqslant (1 - x_1) + x_2.$$

Writing down all four non-expansiveness conditions and simplifying, we get:

Proposition 6.2. *Two pairs $(x_1, x_2), (y_1, y_2) \in \{0,1\}^2$ satisfy the X-gate relation $y_1 y_2 = X(x_1 x_2)$ if and only if they satisfy the following linear system:*

$$\text{(X)} \qquad \begin{aligned} y_1 &= x_1 - \Delta, \\ y_2 &= x_2 - \Delta, \end{aligned}$$

with

$$\Delta \geqslant \max(0, x_1 + x_2 - 1)$$

From the results in the last two sections, we obtain:

Theorem 6.3. (i) *For n individuals with preference lists (over the set of individuals) of total length m, the set of people that are matched in stable matchings can be found in* $O(\sqrt{m}\log^3 m)$ *time on an* m^3*-processor CRCW PRAM.*

(ii) *If a stable matching exists, then it can be found in* $O(\sqrt{m}\log^3 m)$ *time on an* m^4*-processor CRCW PRAM.*

(iii) *A characterization of all the stable matchings by means of a 2-SATISFIABILITY instance can be found within the bounds in* (ii).

References

[1] T. Feder, Stable Networks and Product Graphs, Doctoral Dissertation, Stanford University, 1991.
[2] A.V. Goldberg, S.A. Plotkin, D.B. Shmoys, É. Tardos, Using interior-point methods for fast parallel algorithms for bipartite matching and related problems, SIAM J. Comput. 21 (1992) 140–150.
[3] D. Gusfield, R.W. Irving, The Stable Marriage Problem: Structure and Algorithms, MIT Press Series in the Foundations of Computing, MIT Press, Cambridge, MA, 1989.
[4] M. Kojima, S. Mizuno, A. Yoshise, A polynomial-time algorithm for a class of linear complementarity problems, Math. Program. 44 (1989) 1–26.
[5] E. Mayr, A. Subramanian, The complexity of circuit value and network stability, J. Comp. Systems Sci. 44 (1992) 302–323.
[6] N. Megiddo, Pathways to the optimal set in linear programming, in: N. Megiddo (Ed.), Progress in Mathematical Programming: Interior-Point and Related Methods, Springer, New York, 1988, pp. 131–158.
[7] R.D.C. Monteiro, I. Adler, Interior path following primal-dual algorithms, part I: Linear programming, Math. Program. 44 (1989) 27–41.
[8] A. Subramanian, A new approach to stable matching problems, Technical Report STAN-CS-89-1275, Dept. of Computer Science, Stanford University, 1989.