JOURNAL OF COMPUTER AND SYSTEM SCIENCES 45, 233–284 (1992)

A New Fixed Point Approach for Stable Networks and Stable Marriages

TOMÁS FEDER*

IBM Research, Almaden Research Center,
650 Harry Road, San Jose, California 95120

Received August 5, 1989

In a network stability problem, the aim is to find stable configurations for a given network of boolean gates. For general networks, the problem is known to be computationally hard. Mayr and Subramanian introduced an interesting class of networks by imposing fanout restrictions at each gate, and showed that network stability on this class of networks is still sufficiently rich to express as special cases stable marriage and stable roommate problems. In this paper we study the sequential and parallel complexity of network stability on networks with restricted fanout. Our approach builds on structural properties of these networks and exposes close ties with the theory of retracts and isometric embeddings for product graphs. This structure then gives new efficient algorithms for questions of representation, enumeration, and optimization in stable matching.

1. INTRODUCTION

In the study of problems on boolean circuits, several special cases have been considered. Sometimes the topology of the circuit is restricted by assuming, for example, that the circuit is planar. The second approach limits the computational power of each gate in the circuit by allowing, say, only monotone gates. No restrictions are usually placed on the number of copies of its output value that each gate can produce, and it is a priori unclear whether such a restriction can be imposed in a meaningful way: if each gate produces only two copies of its output value, then an unlimited number of copies can still be obtained by adding more gates; if each gate produces only one copy of its output value, then the circuit reduces to a tree.

It seems then that the usual circuit model is not flexible enough for the study of the output behaviour of boolean gates. Mayr and Subramanian [36] generalized the model by allowing gates that produce several *different* output values. Now it is possible to forbid gates that output several copies of the same value and still obtain non-trivial circuits. Mayr and Subramanian showed that a set of gates lacks the ability to make multiple copies of the same value if and only if each gate in the set is *adjacency-preserving*, i.e., a change in the value of one of the boolean inputs to the gate affects *at most one* of its outputs. One of the simplest such gates is the comparator: it takes two boolean inputs x and y, and produces two outputs $\min(x, y)$

* Work done while the author was at Stanford University.

0022-0000/92 $5.00

and $\max(x, y)$. Note that a gate constructed by combining several smaller adjacency-preserving gates will also be adjacency-preserving.

A *network* is a boolean circuit with feedback. A configuration of a network is an assignment of boolean values to the edges of the network. The configuration is stable if each gate, when evaluated using the values assigned to its incoming edges as inputs, produces as outputs the values assigned to its outgoing edges. The problem of deciding whether a given network has a stable configuration is known to be $\mathcal{NP}$-complete in general, and therefore much harder than the problem of evaluating a boolean circuit, which is only $\mathcal{P}$-complete. In this paper we show that if only adjacency-preserving gates are allowed, then the network stability problem can be solved in polynomial time. The algorithm finds a stable configuration, if one exists, in $O(m^3)$ time, where m is the number of edges in the network. An earlier algorithm of Mayr and Subramanian for a special class of adjacency-preserving gates, called *scatter-free* gates, runs in $O(m)$ time. Our result completes the classification of network stability problems according to the types of gates allowed; when making copies of values is allowed, network stability is in $\mathcal{P}$ for monotone and for linear gates, otherwise $\mathcal{NP}$-complete [36].

We further show that in fact, given a non-monotone set of adjacency-preserving gates, the network stability problem (both as a search problem and as a decision problem) over this set of gates has surprisingly the same parallel complexity as the circuit value problem over the same set of gates, up to $\mathcal{NC}^1$ reductions. This provides further evidence towards the thesis that adjacency-preserving circuit value problems and network stability problems define "robust" parallel complexity clases below $\mathcal{P}$ [36].

The main idea in our approach is to view the stable configurations of an adjacency-preserving network as fixed points of an *edge-preserving* mapping on a hypercube. Such mappings were previously studied by Bandelt, Vel [2, 3], Hell, Nowakowski, Rival [28, 29, 40], Quilliot [43] and others. We use the median structure of the hypercube [37] to show that (a) the set of stable configurations has a simple characterization as a 2-SAT instance on the boolean variables associated with the edges of the network, and (b) the behavior of the network is closely related to a certain permutation on these boolean variables. This leads to efficient algorithms for questions of representation, enumeration, and optimalizations on adjacency-preserving networks.

Two well-known problems that can be studied in this framework are the stable marriage problem and its non-bipartite version, the stable roommate problem (see [15, 35, 41] for an introduction). Subramanian [45] showed that these two problems can be viewed as network stability problems over two specific adjacency-preserving (actually scatter-free) gates, the comparator and the X-gate, respectively. The stable marriage or roommate assignments correspond here to the stable configurations of the network. Some of the structure of adjacency-preserving networks mentioned above has been previously observed and used in the special case of stable matching [21, 22, 31–33]; these results can also be found in [24]. We apply our general results on adjacency-preserving network stability to the special

case of stable matching. For instance, with n people and $m \leqslant \binom{n}{2}$ candidate pairs (pairs of people who list each other in their preference lists), we obtain:

1. An $O(m)$ time algorithm for finding a 2-SAT instance that characterizes the set of stable roommate assignments. From this characterization, all stable roommate assignments can be enumerated in $O(n)$ time per assignment, using $O(m)$ space. This extends the known marriage bounds to the roommates case [21]. Furthermore, the stable pairs for a *fixed* person in a roommate instance can be obtained in $O(m)$ time. Earlier algorithms of Gusfield [22] find the stable pairs for a fixed person in $O(m \log n)$ time, and obtain in $O(nm \log n)$ time a succinct characterization from which all stable assignments are enumerated in $O(m)$ time per assignment using $O(m)$ space.

2. An algorithm for the *weighted* optimal stable marriage problem which runs in $O(\min(n, \sqrt{K})\, m \log(K/m+2))$ time, where K is the weight of an optimal solution. An algorithm for this problem that runs in $O(m^2 \log m)$ time in the unit cost model was given by Irving, Leather, and Gusfield [32]. The unweighted version of the problem, also known as the *egalitarian* stable marriage problem, has $K \leqslant m$ and can therefore be solved in $O(m^{1.5})$ time. An $O(m^2)$ algorithm was known for this problem [32]. Similarly, the *lexicographic* stable marriage problem can be solved in $O(n^{0.5}m^{1.5})$ time by solving n instances with $K \leqslant m/n$ on the average. Here an $O(nm^2 \log^2 n)$ algorithm was known [32]. In general, if $K = O(m^c)$ for some constant c, then our algorithm runs in at most $O(nm \log m)$ time. This is the case for the *parametric* egalitarian stable marriage for a *fixed* value of the parameter λ, which has $K \leqslant 2m^3$. Gusfield and Irving [23] give an algorithm for the parametric optimal stable marriage problem that finds a solution for *all* values of λ in the time it takes to find a solution for a *single* value of λ using their algorithm; an interesting question is then whether a similar result can be obtained with the algorithm presented here.

3. A proof that the *egalitarian* stable roommates problem is $\mathcal{NP}$-complete. This answers an open question of Gusfield [20] and exhibits a question for which the marriage and the roommate cases have a strikingly different complexity (see also [44]). We also obtain an approximation algorithm for the more general *weighted* optimal stable roommate problem which guarantees an approximation factor of 2 and runs in $O(m \log(n^2/m))$ time. Gusfield and Pitt [27] have independently obtained an $O(m^2)$ algorithm with the same approximation guarantee.

4. A construction showing that every 2-SAT instance with n variables and m clauses characterizes the set of solutions of *some* small stable roommates problem with $O(n)$ people and $O(m)$ candidate pairs. the same result holds for stable marriages if the 2-SAT instance is *bipartite* (that is, the graph with edges between every pair of literals that cannot be simultaneously satisfied, is bipartite). An earlier construction of Gusfield, Irving, Leather, and Saks [25, 33] for the marriage case gives instances with $O(m)$ people. The bound given here implies that the egalitarian stable marriage problem with n people and m candidate pairs is at least as hard as

finding the size of a maximum matching on bipartite graphs with n vertices and at most m edges, where vertices may have integer weights adding up to at most m.

5. An $\mathcal{NC}^1$ reduction from the problem of constructing a solution (or even a description of all solutions) to a stable roommates problem to the problem of deciding whether such a solution exists. This shows that stable roommates, even as a search problem, is in the class $\mathscr{CC}$ containing stable marriage, comparator circuit value, and lexicographically first matching as complete problems [36].

Note. Our earlier time bound of $O(m)$ for enumerating *all* stable pairs of a roommate instance [11] was incorrect.

The results extend to networks of adjacency-preserving gates that are not necessarily boolean, through a study of edge-preserving mappings on arbitrary cartesian products of graphs, and of the retracts of such graphs. We generalize a characterization of retracts obtained by Chung, Graham and Saks in their study of dynamic search in graphs [5, 6], and then use a result of Graham and Winkler on isometric embeddings of graphs [19, 48] to obtain a polynomial-time network stability algorithm. Winkler [48] observed that for certain pursuit problems and dynamic location problems, a solution that applies to some collections of graphs can often be extended to the products and retracts of these graphs; this suggests other possible applications for our partial characterization of the retracts of product graphs.

The remainder of this paper is organized as follows. Section 2 gives some basic definitions. Section 3 describes the $O(m^3)$ algorithm for finding a fixed point on the hypercube (and thus a stable configuration for an adjacency-preserving network). Section 4 presents the main structural properties of edge-preserving mappings on the hypercube. Section 5 constructs a small certificate for mappings without fixed point. Section 6 describes an efficient algorithm that finds a 2-SAT characterization for the set of stable configurations. Section 7 summarizes sevaral algorithmic results on the solution space of a 2-SAT instance. Section 8 gives applications to stable matching problems and shows how earlier results on the structure of these problems can be interepreted in the framework presented here. Section 9 presents parallel complexity results. Section 10 extends the results to the non-boolean case, based on the theory of isometric embeddings of Graham and Winkler [19], and Section 11 uses this structure to prove a result on the existence of fixed products. Section 12 explores directions for further research.

Much of the algorithmic material on 2-SAT mentioned in Section 7 is described in full detail in Feder [12] and may have applications outside network stability. The structural properties of 2-isometric subgraphs of cartesian products (a generalization of 2-SAT to the non-boolean case) mentioned in Section 10 are described in greater detail in [13].

Remark. A simpler and more general formulation of the framework presented here appears in [14]. Some of the results obtained include a faster fixed point algorithm for which an $O(m^2)$ time bound can be derived, by means of the notion of

a convergent network [14, 46]; and an $\Omega^*(m^{1/3})$ parallel lower bound for circuit value in the monotone scatter-free model, compared with the known $O^*(m^{1/2})$ upper bound [36]. The notion "adjacency-preserving" is generalized to "nonexpensive" and then applied to arbitrary finite metric spaces. The simpler fixed point algorithm carries over directly to the case of products of finite metric spaces; from a structural point of view, an analogue of the isometric representation theorem [19, 47] for retracts is obtained.

2. Definitions

A λ-input, μ-output *gate* is a function $g: \{0,1\}^\lambda \to \{0,1\}^\mu$ from λ-bit input words to μ-bit output words. In particular, the constant values 0 and 1 are produced by 0-input, 1-output gates. A *network* is a finite labelled directed graph. Each node is labelled with a gate, an ordering of its incoming edges, and an ordering of its outgoing edges. If a node has in-degree λ and out-degree μ, its gate has λ inputs and μ outputs. The edges of the network are labelled $1, 2, \ldots, m$, in some arbitrary order. We refer to the outputs of 0-input gates in a network as the *inputs* of the network, and to the inputs of 0-output gates as the *outputs* of the network. We shall always assume that the constants 0 and 1 are available as inputs to the network. The network is a *circuit* if its directed graph is acyclic.

A *configuration* of a network N is an assignment of boolean values (0s and 1s) to the edges of N, and is represented by an m-bit word $x_1 x_2 \cdots x_m$, where x_i is the value assigned to edge i. We associate with the network N a *transition function* $f_N: \{0,1\}^m \to \{0,1\}^m$ on the set of configurations. This function maps a configuration $x = x_1 x_2 \cdots x_m$ to the configuration $y = y_1 y_2 \cdots y_m$ obtained by evaluating all the gates of N in parallel using the x_i values as inputs and letting the y_i be the resulting output values. We can now view the entire network as a single large gate f_N whose ith output edge feeds into its ith input edge. A configuration x is *stable* if it is invariant under this transition function, that is, if it satisfies the equation $f_N(x) = x$. A binary word x satisfying this equation is called a *fixed point* of f_N.

Two binary words of the same length are *adjacent* if they differ in a most one bit position. We denote by e_i the word x with $x_j = 0$ for $j \neq i$ and $x_i = 1$. Given two words x and y, we denote by $x \oplus y$ the bitwise exclusive-or of x and y. Thus a word x is adjacent to x and to $x \oplus e_i$ for all $1 \leqslant i \leqslant m$. A gate g is *adjacency-preserving* if it maps adjacent input words to adjacent output words. The link between this property and the notion of fanout is the result that a set of gates has the capability to "make copies" if and only if it contains a gate that does not preserve adjacency (see [36]).

We call a network N *adjacency-preserving* if all its gates are adjacency-preserving. This, in turn, is equivalent to requiring that the transition function f_N, viewed as a single large gate, be adjacency-preserving. We can describe this property in graph-theoretic terms. The graph with $M = 2^m$ vertices corresponding to the binary words of length m and with edges between adjacent words is a *reflexive hypercube*, the m-cube (reflexive here means that the graph has a loop at each vertex; this is

consistent with our definition of adjacency). If f_N is adjacency-preserving, it maps adjacent vertices to adjacent vertices in the hypercube and it is therefore an *edge-preserving* mapping. Such a mapping is also called a *homomorphism*; when, in addition, the mapping is one-to-one and onto, it is an *isomorphism*. We denote by $\text{dist}(x, y)$ the distance between two vertices in a graph: this is the Hamming distance in the case of a hypercube. We shall use the following basic property:

LEMMA 2.1. *A mapping f on a reflexive graph is edge-preserving if and only if* $\text{dist}(f(x), f(y)) \leqslant \text{dist}(x, y)$ *for all vertices x and y in the graph. In particular, if a is a fixed point of f, then* $\text{dist}(f(x), a) \leqslant \text{dist}(x, a)$ *for all x.*

Proof. The *if* part is clear: if x and y are adjacent, then $\text{dist}(x, y) \leqslant 1$, so $\text{dist}(f(x), f(y)) \leqslant 1$; hence $f(x)$ and $f(y)$ are adjacent, and the mapping f is edge-preserving. For the *only-if* part, if f is edge-preserving and the vertices x and y are at distance d, then there is a parth of length d from x to y. The images of the vertices along this path form a path of length d from $f(x)$ to $f(y)$, because consecutive vertices on a path are adjacent vertices that must be mapped to adjacent vertices. Therefore $\text{dist}(f(x), f(y)) \leqslant d = \text{dist}(x, y)$. If $y = a$ is a fixed point, then $f(y) = a$ and $\text{dist}(f(x), a) \leqslant \text{dist}(x, a)$ follows as a special case. ∎

We sometimes view edge-preserving mappings as defined on edges as well as vertices. Thus for an edge $e = (x, y)$ we have $f(e) = (f(x), f(y))$. The image of a graph H is then the graph $f(H)$ whose vertices and edges are the images of vertices and edges of H under f.

In Section 4 we shall study structural properties of edge-preserving mappings on the hypercube. For the fixed point algorithm in the next section, however, the characterization given by the preceding lemma is sufficient. Note that if that algorithm is applied to the transition function f_N, the resulting fixed point will be a stable configuration of the network N.

Throughout this paper, whether we view a network as a single large gate or as several small gates, we shall make the complexity assumption, for adjacency-preserving gates, that (a) a gate g is given by an *oracle* which produces an output word given an input word in time proportional to the total input and output sizes, and (b) given a bit position i, the oracle can change the ith bit of its previous input query x and output the bit position (if any) in which $g(x)$ and $g(x \oplus e_i)$ differ, in constant time. This complexity assumption is consistent under the replacement of a single gate by several smaller separate gates.

We will sometimes use a special parameter for a network, called the *gatewidth* of the network, which is the maximum over all gates g in the network of $\min(\lambda, \mu)$, where λ and μ are the number of inputs and the number of outputs of g, respectively.

3. A FIXED POINT ALGORITHM ON THE HYPERCUBE

Given an oracle for an edge-preserving mapping f on the m-cube, our goal is to find a fixed point of f, if one exists. The algorithm will simply maintain a current

point x and its image $y = f(x)$, updating x until it reaches a fixed point. Since $\text{dist}(f(x), a) \leqslant \text{dist}(x, a)$ for all fixed points a, we could try to repeatedly replace x by $f(x)$ with the hope of decreasing $\text{dist}(x, a)$. Unfortunately, during this iterative process, it may turn out that $\text{dist}(f(x), a) = \text{dist}(x, a)$ for exponentially many steps without repeating the same x twice and with the algorithm not knowing that $\text{dist}(x, a)$ is not decreasing. So instead, we measure progress mainly by the decrease in the distance $\text{dist}(x, f(x))$; if $\text{dist}(x, f(x))$ reaches zero, then we know that x is a fixed point. Decreasing $\text{dist}(x, a)$ becomes a secondary goal, and we only replace x with $f(x)$ as suggested above if we are unable to decrease $\text{dist}(x, f(x))$ and if we actually know that $\text{dist}(f(x), a) < \text{dist}(x, a)$ for all fixed points a.

In the following procedure, the point x "chases" its image y in an attempt to decrease $\text{dist}(x, y)$. The procedure takes as input a point x, its image $y = f(x)$, and a coordinate i for which $x_i \neq y_i$.

PROCEDURE pursuit (x^0, y^0, i^0) (All coordinates are initially unmarked.)

$x \leftarrow x^0$; $y \leftarrow y^0$; $i \leftarrow i^0$;
loop
  $x \leftarrow x \oplus e_i$; mark coordinate i;
  $y' \leftarrow y$; $y \leftarrow f(x)$;
  if $y = y' \oplus e_j$ for some unmarked coordinate j
  then $\{i \leftarrow j$; **if** $x_i = y_i$ **then** return $(x, y)\}$
  else return "fail";
endloop;

Procedure *pursuit* always terminates within m iterations, since at most m coordinates can be marked. The following lemma says that if the pursuit fails, then it is likely that $\text{dist}(f(x), a) < \text{dist}(x, a)$.

LEMMA 3.1. *If Procedure pursuit, on inputs* (x, y, i) *satisfying the input conditions, yields outputs* (x', y'), *then* $\text{dist}(x', y') < \text{dist}(x, y)$ *with* $y' = f(x')$. *If pursuit fails, then for every fixed point* a *such that* $x_i \neq a_i$ *we have* $\text{dist}(y, a) < \text{dist}(x, a)$.

Proof. The invariant $x_i \neq y_i$ holds at the beginning of each iteration. Thus the assignment $x \leftarrow x \oplus e_i$ always decreases $\text{dist}(x, y)$ by 1. The assignment $y \leftarrow f(x)$ changes at most 1 bit of y (since f is edge-preserving). If this assignment increases $\text{dist}(x, y)$ by 1 again, then $y = y' \oplus e_j$ with $y_j \neq x_j$, so the pair (x, y) is not returned and we either fail or go back to the beginning of the loop with $\text{dist}(x, y)$ the same as in the previous iteration. Therefore the pair (x, y) can only be returned when the assignment $y \leftarrow f(x)$ does not increase $\text{dist}(x, y)$, i.e., if $\text{dist}(x, y)$ has actually decreased during the last iteration.

To prove the second part of the lemma, suppose that there is some fixed point a such that $x_i \neq a_i$ but $\text{dist}(y, a) = \text{dist}(x, a)$ for the initial values of (x, y, i) (note that $\text{dist}(y, a) > \text{dist}(x, a)$ is not possible by Lemma 2.1). We show that the invariants $\text{dist}(y, a) = \text{dist}(x, a)$ and $y_i = a_i$ (or equivalently, $x_i \neq a_i$, by the invariant $x_i \neq y_i$) must hold at the beginning of each iteration. If the invariants hold at the beginning of some iteration, then the assignment $x \leftarrow x \oplus e_i$ decreases $\text{dist}(x, a)$ by 1 and

forces $\text{dist}(y, a) > \text{dist}(x, a)$, so the assignment $y \leftarrow f(x)$ must decrease $\text{dist}(y, a)$ by 1 as well (by Lemma 2.1 again). This means that after y is updated the invariant $\text{dist}(y, a) = \text{dist}(x, a)$ holds again, and also $y_j = a_j$ with j given by $y = y' \oplus e_j$. So if the end of the loop is reached, the invariant $y_i = a_i$ is also preserved. We therefore have $y_i = a_i$ each time a coordinate i is maked, or $y'_i = a_i$ after the assignment $y' \leftarrow y$, for every marked coordinate i. But since the assignment $y \leftarrow f(x)$ must decrease $\text{dist}(y, a)$ by 1, we must have $y'_j \neq a_j$, so j cannot be a marked coordinate, i.e., we do not return "fail." So the pursuit can only fail if the assumption was false, i.e., if every fixpoint a such that $x_i \neq a_i$ satisfies $\text{dist}(y, a) < \text{dist}(x, a)$. ▮

The main algorithm maintains, together with the current x, a parameter α which bounds $\text{dist}(x, a)$ for every fixed point a.

ALGORITHM fixed point.

pick any x; $y \leftarrow f(x)$; $\alpha \leftarrow m$;
while $x \neq y$ and $\alpha > 0$ **do**
  execute pursuit (x, y, i) for some coordinate i such that $x_i \neq y_i$;
  if *pursuit* outputs a pair (x', y')
  then $x \leftarrow x'$; $y \leftarrow y'$; $\alpha \leftarrow m$
  else if $\text{dist}(x, y)$ is *odd* **then** $x \leftarrow y$ **else** $x \leftarrow y \oplus e_i$;
    $y \leftarrow f(x)$; $a \leftarrow \alpha - 1$;
endwhile;
if $x = y$ **then** "x is a fixed point of f"
  else "f has no fixed point."

Suppose that a is a fixed point. If $d(x, y)$ is odd, then $d(x, a) + d(a, y)$ is odd, so $d(x, a) \neq d(y, a)$, implying that $d(y, a) < d(x, a)$ by Lemma 2.1. This justifies the reduction in the bound α after the assignment $x \leftarrow y$. If $d(x, y)$ is even, then $d(x, a) + d(a, y)$ is even, and we have two cases. If $x_i = a_i$, then $d(y \oplus e_i, a) < d(y, a)) \leqslant d(x, a)$. If instead $x_i \neq a_i$, and *pursuit* fails, then $d(y, a) < d(x, a)$ by Lemma 3.1, and in fact $d(y, a) + 1 < d(x, a)$ by the parity condition, so $d(y \oplus e_i, a) \leqslant d(y, a) + 1 < d(x, a)$. In either case, the reduction in the bound α after the assignment $x \leftarrow y \oplus e_i$ is justified.

LEMMA 3.2. *During the fixed point algorithm, the parameter α is always an upperbound on* $\text{dist}(x, a)$ *for every fixed point a. Furthermore, each iteration of the* while *loop decreases the pair* $(\text{dist}(x, y), \alpha)$ *lexicographically.*

Proof. The upperbound $\alpha = m$ holds vacuously at all times, and we just argued that if the call to *pursuit* fails for x, y, and i, then after the assignment to x the bound α can be justifiably reduced. To prove the second part of the lemma, note that if *pursuit* outputs a pair (x', y') then $\text{dist}(x', y') < \text{dist}(x, y)$ by Lemma 3.1, so the first component of the pair $(\text{dist}(x, y), \alpha)$ is reduced. If *pursuit* fails, then the pair is replaced with $(\text{dist}(x'', y''), a - 1)$, where x'' lies on a shortest path from x to y, and $y'' = f(x'')$. Therefore $\text{dist}(x'', y'') \leqslant \text{dist}(x'', y) + \text{dist}(y, y'') = \text{dist}(x, y) - \text{dist}(x, x'') + \text{dist}(y, y'') \leqslant \text{dist}(x, y)$ by Lemma 2.1, so the first component does not

increase, while the second component is reduced. In either case, the pair decreases lexicographically. ▮

This lemma implies the correctness of the agorithm: If $\alpha = 0$ and we are not at a fixed point, then there is no fixed point. The lemma also bounds the running time of the algorithm: The body of the while loop is executed at most m^2 times, since every iteration decreases the pair $(\text{dist}(x, y), \alpha)$ lexicographically; each iteration involves one calls to *pursuit*; and *pursuit* runs in $O(m)$ time. This gives a total time bound of $O(m^3)$.

THEOREM 3.1. *Given an edge-preserving mapping f on the m-cube, the fixed point algorithm finds a fixed point of f, or proves that none exists, in $O(m^3)$ time.*

Note. If f has no fixed point, then a more complicated algorithm, using the structure of periodic points and fixed cubes developped in the next section, finds a point x such that $\text{dist}(x, f(x))$ is minimum, with x a periodic point as well, within the same time bound.

4. THE STRUCTURE OF EDGE-PRESERVING MAPPINGS

Hypercubes are median graphs [2, 37]. This means that given three points a, b, c in the m-cube, there exists a unique point p which lies simultaneously on a shortest path between a and b, on a shortest path between a and c, and on a shortest path between b and c. The point p is called the *median* of a, b, c, denoted by $\text{med}(a, b, c)$. The coordinates p_i of the median p can be obtained by applying the majority function to the corresponding coordinates a_i, b_i, and c_i.

Let $f^{(k)}$ be the kth iterate of f. Thus $f^{(0)}(x) = x$ and $f^{(k+1)}(x) = f(f^{(k)}(x))$. A point x is *periodic* for f if $f^{(r)}(x) = x$ for some $r > 0$; the least such r is the *period* of x. Thus fixed points are periodic points of period 1. The set of periodic points is called the *attractor set* of f, because for all points x, if k is sufficiently large, then $f^{(k)}(x)$ is in this set.

An edge-preserving map g on a graph G is called a *retraction* if $g^{(2)} = g$. The image $g(G)$ of G under a retraction g is called a *retract* of G. Retracts have received considerable attention in the literature (see [2, 3, 28, 29, 40, 43]). Note that the retracts of a connected graph G are connected subgraphs of G, because if P is a path in G connecting two vertices a and b of the retract $g(G)$, then the image $g(P)$ is a path connecting a and b in $g(G)$. In fact, if P is a shortest path in G, then $g(P)$ is also a shortest path in G, so retracts are *isometric* subgraphs, i.e., the distance between two vertices of $g(G)$ in the graph G is the same as their distance in $g(G)$. If f is any edge-preserving map on a graph G with M vertices, then $g = f^{(M!)}$ is a retraction, because $f^{(k)}(x)$ is a periodic point of f for $k \geqslant M - 1$, and if x is a periodic point of f then $f^{(M!)}(x) = x$ since the period of x is at most M. The vertices of the retract $g(G)$ are the periodic points of f.

LEMMA 4.1. *If a and b are periodic points of an edge-preserving mapping f, then $\text{dist}(f(a), f(b)) = \text{dist}(a, b)$. If a, b, and c are fixed points of an edge-preserving*

mapping f on the m-cube, then $\mathrm{med}(a, b, c)$ *is also a fixed point of f. If a, b, and c are periodic points of f, then* $\mathrm{med}(a, b, c)$ *is also a periodic point of f.*

Proof. If a and b are periodic points, and we let $g = f^{(M!)}$ be the retraction mentioned above, then $g(a) = a$ and $g(b) = b$, so $\mathrm{dist}(a, b) = \mathrm{dist}(g(a), g(b)) \leqslant \mathrm{dist}(f(a), f(b)) \leqslant \mathrm{dist}(a, b)$ by Lemma 2.1, hence dist $(f(a), f(b)) = \mathrm{dist}(a, b)$. To prove the second claim, let a, b, and c be fixed points of f. By definition, the point $p = \mathrm{med}(a, b, c)$ lies on a shortest path between a and b, so Lemma 2.1 gives $\mathrm{dist}(a, b) \leqslant \mathrm{dist}(a, f(p)) + \mathrm{dist}(f(p), b) \leqslant \mathrm{dist}(a, p) + \mathrm{dist}(p, b) = \mathrm{dist}(a, b)$; i.e., the point $f(p)$ also lies on a shortest path between a and b. The same argument applied to the two points a, c and to the two points b, c proves that $f(p)$ lies on a shortest path between a and c and on a shortest path between b and c as well. Thus $f(p)$ is the median of a, b, and c, and so $f(p) = p$. The last claim follows from the fact that the periodic points of f are the fixed points of the retraction $f^{(M!)}$. ∎

A set of points S in the m-cube is a *median set* if the median of every triple of points in S is also in S. Median sets are important here because, by the preceding lemma, both the attractor set and the fixed point set are median sets; in fact, the attractor set is a connected median set, i.e., the subgraph H of the hypercube induced by the vertices in S is connected. (Connectivity follows from the fact that the attractor set is the set of vertices of a retract.) A connected graph induced by the vertices of a connected median set is a *median graph* (and a *median subgraph* of the hypercube).

We would like to have a succinct representation for median sets. Consider an instance I of 2-SAT, i.e., a set of clauses with two literals per clause, on the boolean variables $x_1, \ldots, x_m$. Each clause is of the form $u \vee v$, where each of the literals is either a positive literal x_i or a negative literal $\overline{x_i}$. A *solution* for I is an assignment of values to the boolean variables x_i that makes all the clauses in I true. A variable x_i is *trivial* in I if it has the same value in every solution of I. Two nontrivial variables x_i and x_j (with $i \neq j$) are *equivalent* in I if either $x_i = x_j$ in every solution to I, or $x_i = \overline{x_j}$ in every solution to I. If we interpret the solutions for I as vertices $x = x_1 x_2 \cdots x_m$ in the m-cube, we obtain the following:

LEMMA 4.2. *A set of points in the m-cube is a median set if and only if it is the set of solutions to an instance of* 2-SAT. *Furthermore, the subgraph induced by a median set is connected if and only if the corresponding instance of* 2-SAT *has no equivalent variables.*

Proof. Let S be the set of solutions to an instance I of 2-SAT. We claim that S is a median set. Suppose a, b, and c are in S, and let $p = \mathrm{med}(a, b, c)$. Let C be any clause of I, and let x_i, x_j be the two boolean variables in the clause C. Since p_i is the majority of a_i, b_i, and c_i, p_i is different from at most one of these three values. Similarly, p_j is different from at most one of a_j, b_j, and c_j. Thus $p_i p_j$ is different from at most two of the pairs $a_i a_j$, $b_i b_j$, and $c_i c_j$, and must agree with at least one of these pairs, say $p_i p_j = a_i a_j$. Since a is a solution to I, clause C is true under the assignment $x_i \leftarrow a_i$, $x_j \leftarrow a_j$ and hence under the assignment $x_i \leftarrow p_i$,

$x_j \leftarrow p_j$. Since the clause C was an arbitrary clause of I, it follows that p is a solution to I; i.e., p is also in S and S is a median set.

To prove the converse, let S be a median set in the m-cube. Let I be the set of all the two-literal clauses satisfied by every point in S, and let T be the set of solutions to I. We want to show that $T = S$. Clearly $S \subseteq T$, since every point in S satisfies all the clauses in I. To show that $T \subseteq S$, we pick an arbitrary point a in T and show that a is in S. We will actually show that for all i, j with $1 \leqslant i \leqslant j \leqslant m$, there is a point a^{ij} in S such that $a_k^{ij} = a_k$ for $i \leqslant k \leqslant j$. Since $a^{1m} = a$, the fact that a is in S follows as a special case. The proof is by induction on $j - i$. The base case $j - i = 0$ is easy: since a is in T, the 2-SAT instance I does not contain the clause $x_i \neq a_i$, so this clause is not satisfied by every point in S; i.e., some point a_{ii}^{ii} in S must have $a_i^{ii} = a_i$. To carry out the inductive step, we assume the existence of the two points $a^{i(j-1)}$ and $a^{(i+1)j}$ in S and construct the point a^{ij}. Since a is in T, the 2-SAT instance I does not contain the clause $x_i \neq a_i \vee x_j \neq a_j$, so this clause is not satisfied by every point in S; i.e., some point b in S must have $b_i = a_i$ and $b_j = a_j$. Let $a^{ij} = \text{med}(b, a^{i(j-1)}, a^{(i+1)j})$. Then $a_i^{ij} = a_i$ because $b_i = a_i^{i(j-1)} = a_i$; $a_j^{ij} = a_j$ because $b_j = a_j^{(i+1)j} = a_j$; and $a_k^{ij} = a_k$ for $i < k < j$ because $a_k^{i(j-1)} = a_k^{(i+1)j} = a_k$, completing the induction.

We finally prove the claim about equivalent variables in the statement of the lemma. Let S be the set of solutions to an instance of 2-SAT I and suppose that I has two equivalent variables x_i and x_j, say $x_i = x_j$ for every point in S (the case $x_i = \overline{x_j}$ is identical). Since x_i and x_j are nontrivial variables, there must exist two points a and b in S such that $a_i = a_j = 0$ and $b_i = b_j = 1$. However, no point c in S has $c_i \neq c_j$. It follows that there is no path contained in S from a to b; i.e., the set S is disconnected. To prove the converse, suppose that S is disconnected. Choose two points a and b in different components of the subgraph induced by S, for which $\text{dist}(a, b)$ is minimum. Clearly $\text{dist}(a, b) \geqslant 2$, for otherwise a and b are in the same component. Let i and j be two distinct bit positions such that $a_i \neq b_i$ and $a_j \neq b_j$. We will show that x_i and x_j are equivalent variables of the instance of 2-SAT I characterizing S. Suppose that $a_i = a_j = 0$ and $b_i = b_j = 1$ (the other cases are identical). We claim that every point in S satisfies $x_i = x_j$. For suppose that some point c in S has $c_i \neq c_j$. Then $p = \text{med}(a, b, c)$ has $p_i = c_i$ and $p_j = c_j$. Furthermore, p is in S since S in a median set, $p \neq a, b$ since $p_i \neq p_j$, and p lies on a shortest path from a to b by the definition of median, so that $\text{dist}(a, p)$, $\text{dist}(p, b) < \text{dist}(a, b)$. By minimality of $\text{dist}(a, b)$, the point p must be in the same component as a and in the same component as b, contradicting the fact that a and b are in different components. ∎

The following theorems, whose proofs are built on the two preceding lemmas, give the following picture of the edge-preserving mappings on the m-cube. The first theorem says that, as far as the periodic points are concerned, the mapping is simply given by a permutation σ on the coordinates, that is, $f(x_1 x_2 \cdots x_m) = f(0) \oplus x_{\sigma(1)} x_{\sigma(2)} \cdots x_{\sigma(m)}$ if the all-zero bit vector 0 is taken to be any periodic point. The second theorem says that in fact we do not need to look at the entire m-cube

to find fixed points, and by the third theorem, it is sufficient to look at a subcube contained in the attractor set. Outside the attractor set, the mapping can be quite complicated: we know, for instance, that no finite collection of adjacency-preserving gates yields all edge-preserving mappings under composition. A further illustration of this complexity is given by the number of different edge-preserving mappings. This number is $m^{\Theta(2^m)}$, as opposed to just $2^m m!$ possible mappings within the attractor set. (To see why this bound holds, observe that if we list the 2^m points in some arbitrary order, but with the requirement that every point x listed (except the first one) be adjacent to some previously listed point x', then for every x we have only $m+1$ possible values for $f(x)$, namely the $m+1$ points adjacent to $f(x')$. The lower bound is obtained by considering the mappings that map all even points (points x with an even number of $x_i = 1$) to the same point y and map each odd point to any of the $m+1$ points adjacent to y.) Fortunately, by the fourth theorem, we only need to apply the mapping a polynomial number of times to fall inside the attractor set, where the mapping has a very simple structure.

THEOREM 4.1. *For every edge-preserving mapping f on the m-cube there exists an isomorphism g of the m-cube which coincides with f on the attractor set of f.*

Proof. As we shall see in Section 10, this result can be seen as a special case of a theorem of Graham and Winkler. We give here a direct proof. Assume w.l.o.g. that the attractor set of f contains the all-zero bit vector $x = 0$. Define a permutation σ on $\{1, 2, ..., m\}$ by letting $\sigma(j) = i$ if there are two periodic points a and b such that a and b differ only in bit position i, and $f(a)$ and $f(b)$ differ only in bit position j, and by letting $\sigma(j) = j$ if no such points exist.

We must check that σ is well defined and one-to-one. Suppose that we have two other periodic points a' and b' such that a' and b' differ only in bit position i' and $f(a')$ and $f(b')$ differ only in bit position j'. It is not hard to check that $i = i'$ iff $\text{dist}(a, a') = \text{dist}(b, b')$ and $\text{dist}(a, b') = \text{dist}(b, a')$. Similarly, $j = j'$ iff $\text{dist}(f(a), f(a')) = \text{dist}(f(b), f(b'))$ and $\text{dist}(f(a), f(b')) = \text{dist}(f(b), f(a'))$, and this is in turn equivalent of $\text{dist}(a, a') = \text{dist}(b, b')$ and $\text{dist}(a, b') = \text{dist}(b, a')$ by the first statement in Lemma 4.1. Thus $i = i'$ iff $j = j'$, showing that σ is indeed well defined, one-to-one as far as the first case of the definition of σ is concerned, and trivially one-to-one as far as the second case is concerned. All that is left to check to prove one-to-one-ness is that we cannot have $\sigma(j) = i$ and $\sigma(i) = i$, where $\sigma(j)$ was obtained using the first case of the definition of σ and $\sigma(i)$ was obtained using the second case: indeed, if we have $\sigma(j) = i$, then there are two periodic points a and b that differ only in bit position i, and then the first case of the definition of σ would be used to define $\sigma(i)$ as well by letting $\sigma(i)$ be the bit position in which two periodic points a' and b' such that $f(a') = a$ and $f(b') = b$ differ.

Having defined the permutation σ, we let $g(x_1 x_2 \cdots x_m) = f(0) \oplus x_{\sigma(1)} x_{\sigma(2)} \cdots x_{\sigma(m)}$. Then $g(0) = f(0)$. Suppose that $g(a) = f(a)$ and a is adjacent to b, where a and b are periodic points. Then a and b differ in a single bit position i, $f(a)$ and $f(b)$ differ in the bit position j such that $\sigma(j) = i$ by definition of σ, and $g(a)$ and $g(b)$ also differ in bit position j by definition of g, so that $g(b) = f(b)$ as well. Since the

attractor set is connected, it follows that $g(x)=f(x)$ for all periodic points, as desired. ∎

THEOREM 4.2. *Let f be an edge-preserving mapping on the m-cube, and let S be a connected median subset such that $f(S)\subseteq S$. Then f has a fixed point in the m-cube if and only if f has a fixed point in S.*

Proof. This result can be obtained directly using Lemma 4.2 and Theorem 4.1. We give a simpler proof that uses Theorem 4.3 below. Clearly, if f has a fixed point in S, then f has a fixed point in the m-cube. To prove the converse, suppose that f has a fixed point, say $f(0)=0$. By Theorem 4.3, there is a subcube Q contained in S such that $f(Q)=Q$. Let a be the point in Q closest to 0; this point is unique because Q is a cube. By Lemma 2.1, we must have $\text{dist}(f(a),0)\leqslant\text{dist}(a,0)$. But $f(a)$ is in Q, so by uniqueness $f(a)=a$, and f has a fixed point a in S, as desired. ∎

THEOREM 4.3 (Bandelt and Vel [3]). *For every edge-preserving mapping f on a connected median subset S of the m-cube there exists a subcube Q contained in S such that $f(Q)=Q$.*

A proof of a generalization of this result to products of cliques is given in Section 11.

THEOREM 4.4. *Let f be an edge-preserving mapping on the m-cube, and let x be a point at distance d from the attractor set of f. Then $f^{(k)}(x)$ can only be nonperiodic if $k<dm^2$.*

Proof. Only the case $d=1$ needs to be shown, because the result for general d then follows by induction. If x is at distance $d+1$ from the attractor set of f, then x is at distance d from some point y at distance 1 from the attractor set. Then $f^{(m^2)}(y)$ is in the attractor set by the case $d=1$, so $z=f^{(m^2)}(x)$ is at distance at most d from the attractor set since $\text{dist}(f^{(m^2)}(x), f^{(m^2)}(y))\leqslant\text{dist}(x,y)=d$ by Lemma 2.1. By inductive hypothesis, $f^{((d+1)m^2)}(x)=f^{(dm^2)}(z)$ is a periodic point.

The proof for the case $d=1$ relies on the structure of the attractor set and of the behavior of f in the neighborhood of the attractor set. We first split the set of dimensions $[m]=\{1,\ldots,m\}$ into two disjoint sets V_1 and V_2. The set V_1 consists of those dimensions i such that the attractor set contains both points x with $x_i=0$ and points x with $x_i=1$. The remaining dimensions j are put in V_2, and we shall assume w.l.o.g. that the all-zero bit vector is a periodic point so that $x_j=0$ for all points x in the attractor set and j in V_2. We let m_1 and m_2 be the cardinalities of V_1 and V_2, respectively, so that $m=m_1+m_2$.

If x is at distance 1 from the attractor set, then x is adjacent to some point y in the attractor set along some dimension i. We first consider the case where i is in V_1 and show that $f^{(m_1^2)}(x)$ is a periodic point. For each nonperiodic point $x^k=f^{(k)}(x)$, $k\geqslant 0$, we define an ordered pair $\pi^k=(i^k,j^k)$, where i^k and j^k are both in V_1. We then show that all pairs π^k are distinct. Since there are at most m_1^2 such pairs, there must be at most m_1^2 nonperiodic points x^k, so $f^{(m_1^2)}(x)$ is periodic, as desired.

We now define the pairs $\pi^k = (i^k, j^k)$. Let $y^k = f^{(k)}(y)$. Then y^k is a periodic point and x^k is a nonperiodic point adjacent to y^k along some dimension i^k. We claim that there is a periodic point z^k such that $z^k_{i^k} \neq y^k_{i^k}$ and z^k lies on a shortest path from y^k to every periodic point z such that $z_{i^k} \neq y^k_{i^k}$. The existence of such points z is clear: Since i is in V_1 by assumption, there must exist a periodic point z such that $z_i \neq y_i$; then x is on a shortest path between the periodic points y and z, so x^k is on a shortest parth between the periodic points y^k and $f^{(k)}(z)$, implying that $f^{(k)}_{(z)}$ is a periodic point that differs from y^k in dimension i^k, as desired. Let z^k be a periodic point such that $z^k_{i^k} \neq y^k_{i^k}$ for which dist(y^k, z^k) is minimum. Then if z is another periodic point with $z_{i^k} \neq y^k_{i^k}$, the median $p = \text{med}(y^k, z^k, z)$ is also a periodic point with $p_{i^k} = z^k_{i^k} = z_{i^k} \neq y^k_{i^k}$, and p lies on a shortest path from y^k to z^k, so by minimality of dist(y^k, z^k) we must have $p = z^k$. Since p lies on a shortest path from y^k to z, it follows that z^k lies on a shortest path from y^k to z, as desired. To define j^k, we consider a shortest path in the attractor set from y^k to z^k, and we let t^k and u^k be the last two points before z^k along this path. The point u^k is uniquely determined: it must be the point adjacent to z^k along i^k (otherwise $u^k_{i^k} \neq y^k_{i^k}$ and so z^k would have to lie on a shortest path from y^k to u^k, a contradiction). The point t^k can be any periodic point adjacent to u^k and on a shortest path from y^k to u^k, and we will only restrict the choice of such a t^k if $u^k = f(u^{k-1})$, in which case we will always pick $t^k = f(t^{k-1})$. Now j^k is defined to be the dimension along which t^k differs from u^k. We shall also need later on the point s^k which is adjacent to z^k along j^k and to t^k along i^k. The points z^k, u^k, t^k, and s^k form a square; the first three of these four are periodic points, while s^k is nonperiodic because u^k is the only periodic point adjacent to z^k on a shortest path from y^k to z^k.

Having defined the pairs $\pi^k = (i^k, j^k)$, we only need to prove that they are all distinct. The key idea in the proof is the fact that if a pair repeats, say $\pi^k = \pi^{k'}$ with $k < k'$, then for all l such that $k \leqslant l < k'$, if s is a nonperiodic point adjacent to two periodic points along dimensions i^l and j^l, respectively, then the image $f(s)$ is a nonperiodic point adjacent to two periodic points along dimensions i^{l+1} and j^{+1}, respectively. Using this fact inductively, starting with the nonperiodic point s^k adjacent to two periodic points along dimensions i^k and j^k, we conclude that $f^{(r)}(s^k)$ is a nonperiodic point adjacent to two periodic points along dimensions $i^{k+(r \bmod (k'-k))}$ and $j^{k+(r \bmod (k'-k))}$ for all $r \geqslant 0$, since $i^{k'} = i^k$ and $j^{k'} = j^k$. But this means that $f^{(r)}(s^k)$ remains nonperiodic for all $r \geqslant 0$, which is impossible.

To prove the key fact, we associate with each pair $\pi^k = (i^k, j^k)$ a quantity c^k which is monotonically nondecreasing with k. Thus we can only have $\pi^k = \pi^{k'}$ if $c^k = c^{k'}$, i.e., only if all c^l with $k \leqslant l \leqslant k'$ are equal. The quantity c^k is simply the number of perioidc points w such that $w_{j^k} = b^k$, where the binary value b^k must be chosen so that all these periodic points w agree on the value of w_{j^k}. The choice of b^k and the value of c^k only depend on the pair π^k and not on k itself; furthermore, the choice of b^k is such that $z^k_{j^k} = b^k$: if we pick $b^k \neq z^k_{j^k}$ we have two periodic points t^k and u^k which satisfy $w_{j^k} = b^k$ but do not agree on w_{j^k}, contrary to the condition on b^k; on the other hand, if we pick $b^k = z^k_{j^k}$, then a periodic point w which satisfies $w_{j^k} = b^k$ has $w_{i^k} \neq y^k_{i^k}$, so z^k lies on a shortest path from y^k to w; therefore $w_{j^k} = z^k_{j^k}$

and the consistency condition on b^k is met. To prove the monotonicity property for c_k, note that if $w_{i^k} = b^k$, then $w_{i^k} \neq y^k_{i^k}$, so the point x^k lies on a shortest path from y^k to w; therefore x^{k+1} lies on a shortest path from y^{k+1} to $w' = f(w)$; hence $w'_{i^{k+1}} \neq y^{k+1}_{i^{k+1}}$, implying that $w'_{i^{k+1}} = b^{k+1}$. This shows that each periodic w with $w_{i^k} = b^k$ produces a different periodic point $w' = f(w)$ with $w'_{i^{k+1}} = b^{k+1}$, proving that $c^{k+1} \geqslant c^k$ as claimed. We can make a stronger claim and show that, in fact, $c^{k+1} > c^k$ unless $z^{k+1} = f(z^k)$. Suppose that $z^{k+1} \neq f(z^k)$. Recall that $z^k_{i^k} = b^k$ so that the image $z' = f(z^k)$ satisfies $z'_{i^{k+1}} \neq y^{k+1}_{i^{k+1}}$ and z^{k+1} lies on a shortest path in the attractor set from y^{k+1} to $z' = f(z^k)$. The last point on this path before z' must be $f(u^k)$, because the last point on a shortest path in the attractor set from y^k to z^k is necessarily u^k. Since $z^{k+1} \neq z'$, the point z^{k+1} also lies on a shortest path from y^{k+1} to $f(u^k)$. The point $w' = f(w)$ for $w = u^k$ gives us then an extra point w' such that $w'_{i^{k+1}} = b^{k+1}$, but with $w_{i^k} \neq b^k$, so that $c^{k+1} > c^k$.

The result is that if $c^{l+1} = c^l$, then $z^{l+1} = f(z^l)$, in which case $u^{l+1} = f(u^l)$ by uniqueness and $t^{l+1} = f(t^l)$ by definition. Thus $\sigma(i^{l+1}) = i^l$ and $\sigma(j^{l+1}) = j^l$, where σ is as defined in the proof of Theorem 4.1, because the periodic points z^l and u^l, which differ along dimension i^l, have images under f that differ along i^{l+1}, and the periodic points u^l and t^l, which differ along dimension j^l, have images under f that differ along j^{l+1}. If s is a nonperiodic point adjacent to two periodic points z and t along dimensions j^l and i^l, respectively, then there is a fourth point u adjacent to z and t along dimensions i^l and j^l, respctively, and u is a periodic point. Then $u' = f(u)$ is adjacent to $z' = f(z)$ and $t' = f(t)$ along dimensions i^{l+1} and j^{l+1}, respectively, by the values of σ just mentioned. Note that $z_{i^l} = b^l$ because otherwise $u_{i^l} = t_{i^l} = b^l$ for two periodic points u and t that disagree along j^l, contradicting the condition on b^l. Therefore the periodic point z satisfies $z_{i^l} = z^l_{i^l}$. Thus $z_{i^l} \neq y^l_{j^l}$, and there is a shortest path from y^l to x^l to s to z. The image of this path is a shortest path from y^{l+1} to x^{l+1} to $s' = f(s)$ to $z' = f(z)$. Since $x^{l+1}_{i^{l+1}} \neq y^{l+1}_{i^{l+1}}$ and $z'_{i^{l+1}} \neq y^{l+1}_{i^{l+1}}$, the existence of this shortest path implies that $s'_{i^{l+1}} \neq y^{l+1}_{i^{l+1}}$, so $s'_{i^{l+1}} = b^{l+1}$. This shows that $s' \neq u'$, so s' must be adjacent to z' and t' along dimensions j^{l+1} and i^{l+1} instead. Since $s'_{j^{l+1}} \neq z'_{j^{l+1}}$, the consistency condition in the definition of b^{l+1} implies that s' is not a periodic point.

We have thus shown the key fact that if s is a nonperiodic point adjacent to two periodic points along dimensions i^l and j^l, respectively, then the image $s' = f(s)$ is a nonperiodic point adjacent to two periodic points along dimensions i^{l+1} and j^{l+1}, respectively, under the assumption that $c^{l+1} = c^l$. As we stated before, this assumption does hold if $\pi_k = \pi_{k'}$ and $k \leqslant l < k'$, and then the key fact just proved implies that $f^{(r)}(s^k)$ remains nonperiodic for all $r \geqslant 0$, which is impossible. Therefore we must have $\pi_k \neq \pi_{k'}$, i.e., all π_k must be different, so there are at most m_1^2 nonperiodic points x^k and $f^{(m_1^2)}(x)$ is a periodic point, completing the proof in the case where x is a nonperiodic point adjacent to a periodic point along a coordinate in V_1.

The second case that we set out to prove is the case where x is a nonperiodic point adjacent to a periodic point y along a dimension i in V_2. Recall that the all-zero bit vector 0 is a periodic point and that $x_i = 0$ for all periodic points x and i in V_2. This case is somewhat simpler; we will show that for some $k \leqslant (2m_1 + 1)m_2$,

$x' = f^{(k)}(x)$ is either equal to $y' = f^{(k)}(y)$ or adjacent to y' along a coordinate in V_1. In either case, we have that $f^{(m_1^2+k)}(x) = f^{(m_1^2)}(x')$ is a periodic point and $m_1^2 + k \leqslant m^2$, as desired.

To bound the number of iterations through which x remains nonperiodic and adjacent to a periodic point along a dimension in V_2, we define a sequence of distinct 2-SAT clauses $C_1, ..., C_r$, where each clause is of the form $\overline{x_i} \vee u$ with i in V_2, and where either $u = \overline{x_i}$ or u involves a variable x_j with j in V_1. This implies a bound $r \leqslant (2m_1 + 1)m_2$ on the number of possible clauses. We let S be the set of all points that are either periodic or adjacent to a periodic point along a dimension in V_2. The set S is a median set; if x^1, x^2, x^3 are three points in S adjacent to periodic points y^1, y^2, y^3 along dimensions i^1, i^2, i^3, respectively (where $i^c = 0$ if $x^c = y^c$ is periodic, by convention), then $\text{med}(x^1, x^2, x^3)$ is adjacent to the periodic point $\text{med}(y^1, y^2, y^3)$ along the dimension indicated by the majority of i^1, i^2, i^3 (this majority is set to 0 by convention if no majority exists), and therefore $\text{med}(x^1, x^2, x^3)$ is also in S. For every point x in S satisfying $C_1, ..., C_k$, $k \geqslant 0$, the image $f(x)$ will be either a nonperiodic point adjacent to a periodic point along a dimension in V_1, or a point in S satisfying $C_1, ..., C_{k+1}$. Furthermore, if x is a point in S satisfying all the clauses $C_1, ..., C_r$ then x will necessarily be a periodic point. This implies that if x is a point in S, then for some $k \leqslant r$ the point $f^{(k)}(x)$ is either a periodic point or a nonperiodic point adjacent to a periodic point along a dimension in V_1, proving the theorem.

The sequence of clauses is constructed iteratively by letting C_{k+1} be any clause of the restricted form stated above, such that for every x in S satisfying $C_1, ..., C_k$, if $f(x)$ is in S then $f(x)$ satisfies C_{k+1}. We must show that if no such clause exists, then all the points in S satisfying $C_1, ..., C_k$ are periodic points and we can set $r = k$. Let V_3 be the set of all i in V_2 such that $\overline{x_i}$ is not one of the k given clauses. We define a permutation μ on $\{1, ..., m\}$ by letting $\mu(i) = \sigma(i)$ if i is not in V_3, where σ is the permutation used in the proof of Theorem 4.1, and by letting $\mu(i) = j$ if i is in V_3, where j is a dimension in V_3, such that there exists a nonperiodic point x adjacent to a periodic point y along dimension j, the point x satisfies the k given clauses, and $f(x)$ and $f(y)$ are adjacent along dimension i. The function μ is well defined: given i in V_3, it is always possible to find a corresponding j in V_3, because otherwise we could let C_{k+1} be the clause $\overline{x_i}$, since no point in S satisfying the k given clauses would be mapped to a point with the ith coordinate equal to 1. Furthermore, the function μ is one-to-one, because σ is one-to-one from V_1 to V_1 and from $V_2 - V_3$ to $V_2 - V_3$ by definition, and μ must be one-to-one from V_3 to V_3. For if $\mu(i) = \mu(i') = j$ with $i \neq i'$ and both i and i' in V_3, then there exist nonperiodic points x and x', adjacent to periodic points y and y', respectively, along dimension j, such that $f(x)$ and $f(x')$ are adjacent to $f(y)$ and $f(y')$ along dimensions i and i', respectively. This implies $\text{dist}(f(x), f(x')) \leqslant \text{dist}(x, x') = \text{dist}(y, y') = \text{dist}(f(y), f(y')) = \text{dist}(f(x), f(x')) - 2$, a contradiction. Having shown that μ is a permutation, it also follows that μ is uniquely defined, because if we had an alternative choice for j in the definition of $\mu(i)$ for i in S_3, say $j = \mu(i')$, then we could set $\mu(i) = j = \mu(i')$ and μ would not be one-to-one, a contradiction.

Let $h(x_1x_2\cdots x_m)=f(0)\oplus x_{\mu(1)}x_{\mu(2)}\cdots x_{\mu(m)}$. Let T be the set of all points y in S such that $y=f(x)$ for some x in S satisfying $C_1, ..., C_k$. We claim that if y is in T, then $x=h^{-1}(y)$ is an appropriate choice for x given y in the preceding definition of T; we also claim that T is a median set. If y is a periodic point, then $g^{-1}(y)=h^{-1}(y)$ and $f(g^{-1}(y))=y$, where g is the isomorphism defined using σ in Theorem 4.1, so $x=h^{-1}(y)$ is an appropriate choice for x. If y is a nonperiodic point adjacent to a periodic point y' along a dimension i in V_3, then $y=f(x)$, where x is a nonperiodic point satisfying $C_1, ..., C_k$ and adjacent to a periodic point x' with $f(x')=y'$ along a dimension j in V_3. By definition and uniqueness of μ, we have $j=\mu(i)$, and, since $f(x')=g(x')=h(x')$, we obtain $f(x)=h(x)$, or $f(h^{-1}(y))=y$, so $x=h^{-1}(y)$ is an appropriate choice here as well. Now if y^1, y^2, y^3 are in T, then $y=\text{med}(y^1, y^2, y^3)$ is in S (since y^1, y^2, y^3 are in S and S is a median set), y satisfies $f(h^{-1}(y))=y$ (since the median of three fixed points of $f\circ h^{-1}$ is a fixed point of $f\circ h^{-1}$), and $x=h^{-1}(y)$ is in S and satisfies $C_1, ..., C_k$ because $x=\text{med}(h^{-1}(y^1), h^{-1}(y^2), h^{-1}(y^3))$ and each $h^{-1}(y^i)$ is in S and satisfies $C_1, ..., C_k$. Therefore y is in T, and T is a median set. Let I be a 2-SAT instance characterizing T. Suppose that there is a point y in S satisfying $C_1, ..., C_k$ which is not in T. Then y is adjacent to a periodic point y' along a dimension i in V_2 with y' in T, so y must violate some clause $\overline{x_i}\vee u$ of I, providing the required clause C_{k+1} (a clause of the form $\overline{x_i}\vee x_j$ with j also in V_2 and $j\neq i$ can be simplified to just $\overline{x_i}$ because no x in S can have $x_i=x_j=1$; a clause of the form $\overline{x_i}\vee\overline{x_j}$ with j in V_2 and $j\neq i$ cannot be violated by y for the same reason; thus the restrictions on the form of the clause $\overline{x_i}\vee u$ mentioned above are met). If no such clause exists, then every y in S satisfying $C_1, ..., C_k$ is in T, so $f(T)=T$ and all the points in T are periodic points, completing the proof. ▮

The bound $k<m^2$ for $d=1$ is essentially tight. The following example is due to Subramanian [43]. Let $m=2n+1$ and define $y=f(x)$ by $y_i=x_{i-1}$ for $i\neq 1, n+1$, $y_1=x_n\overline{x_{2n+1}}$, $y_{n+1}=x_{2n+1}\overline{x_n}$. Then the point x with $x_i=1$ for $i=1, n+1$, and $x_i=0$ elsewhere is a nonperiodic point adjacent to two periodic points (the two neighbors of x with a single bit set to 1.) If we iterate f on x, the two bits of x set to 1 cycle through the first n and the last $n+1$ bit positions of x, respectively. After $k=n(n+1)-1$ iterations, we obtain the nonperiodic point $x'=f^{(k)}(x)$ with $x'_i=1$ for $i=n$, $2n+1$ and $x'_i=0$ elsewhere, while one more iteration yields a fixed point $f(x')=0$. Thus the m^2 bound cannot be improved below $n(n+1)$.

We have no evidence, on the other hand, that the bound in the last theorem must increase with d. While the stated bound becomes as large as m^3 when d grows, all the examples that we have been able to construct exhibit only quadratic growth. This seems more in accordance with the structure observed throughout this section, since *pairs* of dimensions are almost always involved. Graham [18] has shown that a comparator circuit with m variable inputs has at most $\binom{m}{2}$ nontrivial gates (gate in a circuit is trivial if it can be split into gates with at most one input and one output without affecting the behaviour of the circuit over the set of all possible values for the inputs to the ciruit). His proof can be extended to obtain an $\binom{m+1}{2}-1$

bound on the number of nontrivial gates in a circuit built with gates with at most two inputs per gate. Both bounds are tight, and they imply that if f can be represented by a circuit of gates that have at most two inputs each, then the bound in the preceding theorem can be reduced to $\binom{m+1}{2}$. In fact, we have recently been able to show that a quadratic bound holds in the preceding theorem if f is *scatter-free* [36]. Thus an example achieving super-quadratc growth would require the use of adjacency-preserving gates that are not scatter-free.

Part of the material on retracts, median sets, and 2-SAT described here can be found in [2, 5]. The following sections build efficient algorithms from the structure presented in this section.

5. A Small Certificate for Mappings without Fixed Point

The algorithm of Section 3 shows that by looking at a fairly small fraction of the 2^m points in the hypercube (at most a number polynomial in m) it is possible to decide whether f has a fixed point. Obviously, a single point is enough if f has a fixed point. The work in this section was initiated with the aim of finding a lower bound, i.e., of proving that if f has no fixed point, then it is necessary in some cases to know the images of a fairly large number of points, say some power of m, to certify that f has no fixed points. At some point, however, certificates consisting of only $O(\log m)$ points were obtained. The result in this section gives a final answer of 4. It is easy to see that 4-point certificates are needed for some functions f. For example, if f is the function on the 2-cube given by $f(x_1x_2)=\overline{x_1}\,\overline{x_2}$, then f has no fixed points, but an adjacency-preserving function that agrees with f in three of these four points could still have the fourth one as a fixed point.

THEOREM 5.1. *Let f be an adjacency-preserving mapping on the m-cube with no fixed point. Then there exist four points x^1, x^2, x^3, x^4, such that if f' is an adjacency-preserving mapping on the m-cube satisfying $f'(x^i)=f(x^i)$ for $1\leqslant i\leqslant 4$, then f' has no fixed point.*

Proof. We let the points x^1 and x^2 be any two opposite corners of a minimal fixed cube Q (see Theorem 4.3), and we let the points x^3 and x^4 be another two opposite corners of Q, where x^3 is obtained from x^1 by picking an odd number of odd-type cycles of σ (from Theorem 4.1) and complementing the bits contained in those cycles. The details are as follows.

We assume w.l.o.g. that x^1 is the all-zero bit vector. The fixed cube Q is then the set of all points x that satisfy $x_i=0$ for all i in S, where S is some subset of the coordinates $[m]=\{1,\ldots,m\}$. The coordinates x_i of a point x in Q with i in $T=[m]-S$ can take arbitrary boolean values. The point x^2 opposite to x^1 in Q has $x_i^2=1$ if $i\in T$ and $x_i^2=0$ if $i\in S$. By Theorem 4.1, the mapping f is given for all periodic points by $f(x_1\cdots x_m)=a\oplus x_{\sigma(1)}\cdots x_{\sigma(m)}$, where σ is a permutation on $[m]$ and $a=f(0)$. Let $U\subseteq[m]$ be a cycle of σ. Note that $y_i=a_i\oplus x_{\sigma(i)}$ for all periodic points x and $y=f(x)$. We say that U is an *odd-type cycle* if the number

of coordinates i in U for which $a_i = 1$ (i.e., $y_i = \overline{x_{\sigma(i)}}$ for periodic points) is odd, and an *even-type cycle* if this number is even.

We claim that all cycles U are either contained in T or contained in S; furthermore, all odd-type cycles are contained in T, and all even-type cycles are contained in S. For if $i = \sigma(j)$ is in T, then there are two points x, x' in Q with $x_i \neq x'_i$, so the two points $y = f(x)$ and $y' = f(x')$ in Q satisfy $y_j \neq y'_j$ and j is also in T. So U is contained in either T or S. If U is contained in S then all coordinates i in U satisfy $y_i = x_{\sigma(i)} = 0$ for x in Q and $y = f(x)$, since Q is a fixed cube, so $a_i = 0$ and U is an even-type cycle. So all odd-type cycles are contained in T. Suppose that some even-type cycle U is contained in T. Set $x_i = 0$ for all i in S and assign values to all x_i with i in U so as to satisfy $x_i = a_i \oplus x_{\sigma(i)}$ for all i in the cycle U. These equations can be simultaneously satisfied precisely because the number of complementations, i.e., the number of $a_i = 1$ for i in U, is even. The subcube Q' defined by these constraints is a subcube of Q, and also a fixed cube: if x is in Q' then $y = f(x) = g(x)$ is also in Q', because $y_i = a_i \oplus x_{\sigma(i)} = x_i$ for i in U. The existence of a fixed subcube Q' contained in Q contradicts the minimality of Q. Therefore no even-type cycle can be contained in T, and all even-type cycles are contained in S. Note that this shows, in particular, that Q is a fixed point iff σ has no odd-type cycles.

We can now pick an odd number of odd-type cycles U_l, and let x^3 be the point obtained from x^1 by complementing all the bits in every chosen U_l. This point x^3 is in Q because the bits complemented are all in T. The point x^4 opposite to x^3 in Q is obtained from x^3 by complementing all the bits in T.

Now suppose that $f'(x^i) = f(x^i)$ for $1 \leqslant i \leqslant 4$, and that f' has a fixed point. If z is any point in Q, then z is on a shortest path from x^1 to x^2, and since $\operatorname{dist}(f'(x^1), f'(x^2)) = \operatorname{dist}(x^1, x^2)$, the point $f'(z)$ must be on a shortest path from $f'(x^1)$ to $f'(x^2)$; these two points are in Q, so $f'(z)$ is also in Q. This shows that $f'(Q) \subseteq Q$. By Theorem 4.2, f' must have a fixed point z in Q. We now use the function $g(x_1 \cdots x_m) = f(0) \oplus x_{\sigma(1)} \cdots x_{\sigma(m)}$ that coincides with f on the periodic points of f and define $h = g^{-1} \circ f'$. Then $h(x^i) = x^i$ for $1 \leqslant i \leqslant 4$, and $h(z) = g^{-1}(z)$. Here we use the fact that, since $V = \bigcup U_l$ is the union of an odd number of odd-type cycles, and $h(z) = g^{-1}(z)$, the two points z and $w = h(z)$ cannot have the same number of coordinates i in V set to 1 (an odd number of such coordinates get complemented by g^{-1}). There are now four cases, depending on whether w has fewer 1s or more 1s than z in V, and whether w has at most as many 1s or at least as many 1s as z in $T - V$. Say w has more 1s than z in V, and w has at most as many 1s as z in $T - V$. Then $\operatorname{dist}(w, x^4) > \operatorname{dist}(z, x^4)$, because x^4 has 0s in V and 1s in $T - V$. This means that $\operatorname{dist}(h(z), h(x^4)) > \operatorname{dist}(z, x^4)$, a contradiction. The other three cases are analogous and use the points x^1, x^2, or x^3 instead of x^4. In all cases we reach a contradiction, so the fixed point z cannot exist. ∎

The preceding construction can be used to obtain either two-point or four-point certificates, depending on whether the total number of odd type cycles is odd or even, i.e., on whether the distance between a periodic point and its image is odd or even (if this number is odd, then setting $x^3 = x^2$ and $x^4 = x^1$ is allowed in the

construction). In the odd case the number of points used (two) is optimal, while in the even case it can be shown that any certificate will need at least three points. We do not know whether a function with an optimal certificate of exactly three points does exist.

This result does not say much if we are interested in deterministic lower bounds. While the proof pays special attention to the set of periodic points, it seems that in reality most of the computational complexity of the problem comes from the behaviour of f outside the set of periodic points, where little is known. For example, when a circuit value problem is expressed as a network stability problem [36], the periodic structure is trivial, consisting of just a single fixed point or two adjacent periodic points that get mapped to each other. In both cases these periodic points will provide a certificate of existence or non-existence of fixed points. However, circuit value problems do not seem to be much easier than network stability problems (see [36] and also Section 9).

The oracle model turns out to be quite useful for obtaining deterministic lower bounds on the number of queries of f needed to decide whether f has a fixed point. In parallel complexity, the analogous question allows algorithms that make a number of simultaneous queries of f, that is, polynomial in m, an asks how many such sets of queries are needed to decide whether f has a fixed point. We have recently obtained an $\Omega((m/\log m)^{1/3})$ lower bound on the parallel time complexity of this problem. The bound holds even if randomization is allowed, if f is monotone and scatter-free, and for the easier circuit-value problem [14]. The best known parallel algorithm for evaluating scatter-free circuits is due to Mayr and Subramanian [36] and gives an $O(m^{1/2} \log m)$ upper bound.

6. Efficient Representation of Stable Configurations

We now use the structure from Section 4 to represent the stable configurations of a network efficiently. Suppose that we are given a network with transition function f, as well as a fixed point x of f, say the all-zero bit vector $x = 0$. The idea of the algorithm is then to (a) find the permutation σ associated with the isomorphism g promised by Theorem 4.1 (recall that $g(x_1, ..., x_m) = x_{\sigma(1)} \cdots x_{\sigma(m)}$), and then (b) use σ to find the 2-SAT characterization for the fixed point set guaranteed by Lemma 4.2. This representation can then be used, for example, to enumerate the stable configurations efficiently.

The disjoint cycles of the permutation σ give cycles in the network when we associate the coordinate x_i of a configuration x with the edge i in the network. These cycles in the network are found as follows. Starting with an arbitrary coordinate (i.e., and edge in the network), we build a path of coordinates (edges) in the network. The next coordinate on the path is always found using the *successor* procedure; each new coordinate j on the path is colored *green*, and its predecessor on the path is remembered in pred[j]. If the path dies off because *successor* fails, then all coordinates on the path are re-colored *black* and a new path is started somewhere else. Otherwise, the path must eventually cycle, in which case all coordinates

on the cycle are re-colored *red*; for these red coordinates j, the value of $\sigma(j)$ is then given by pred$[j]$. The green path is then continued from the last green edge on the path. At the end of the algorithm, we only have black coordinates and cycles of red coordinates; if we now set $\sigma(j)=j$ for all black coordinates, we have obtained the permutation σ and hence the isomorphism g.

ALGORITHM cycles.

pick a coordinate i and color i green;
repeat
  execute successor (i);
  if *successor* returns a coordinate j
  then if j is uncolored
    then pred$[j] \leftarrow i$; $i \leftarrow j$; color i green
    else {**if** pred$[j]$ is undefined
      then pred$[j] \leftarrow i$; pick an uncolored i and colore i green (if such an i exists)
      else interchange pred$[j]$ and i;
      follow the pred pointers from j and color all coordinates reached red};
  else follow the pred pointers from i, color all
    coordinates j reached black (including i) and set pred$[j] \leftarrow j$,
    pick a new uncolored i and color i green (if an such i exists);
until all coordinates are either red or black.

The Procedure *successor*, on input i^0, iteratively changes different coordinates i in x until the corresponding coordinate j that changes in $y=f(x)$ is not red, then returns this coordinates j provided it is not a black coordinate. Thus either j is uncolored and the path can be extended, or j is green and a cycle has been closed.

PROCEDURE successor (i^0). (All coordinates are initially unmarked.)

$x \leftarrow 0$; $y \leftarrow f(x)$; $i \leftarrow i^0$;
loop
  $x \leftarrow x \oplus e_i$; $y' \leftarrow y$; $y \leftarrow f(x)$;
  if $y = y' \oplus e_j$ for some unmarked non-black coordinate j
  then if j is red
    then $i \leftarrow$ pred$[j]$; mark j;
    else return j
  else return "*fail*";
endloop;

Let $\sigma(j)$ be the final value of pred$[j]$ produced by the algorithm.

LEMMA 6.1. *If x is a periodic point of f, then $f(x)=g(x)$, where g is the isomorphism of the m-cube given by $g(x_1 \cdots x_m)=x_{\sigma(1)} \cdots x_{\sigma(m)}$.*

Proof. We will show that three invariants are maintained by the algorithm, for all periodic points x and $y=f(x)$: (1) $x_i=0$ if i is black, (2) $x_i \leqslant y_j$ if j is green and pred$[j]=i$; and (3) $x_i=y_j$ if j is red and pred$[j]=i$. At the end of the algorithm,

$\sigma(i) = i$ if i is black because pred$[i] = i$, and $\sigma(j) = i$ if i is red and pred$[j] = i$, so $x_{\sigma(j)} = y_j$ for all j, proving that $f(x_1 \cdots x_m) = x_{\sigma(1)} \cdots x_{\sigma(m)}$.

The invariants hold vacuously at the beginning of the algorithm, since all coordinates are uncolored. Suppose that at some stage successor (i^0) is called, with i^0 green, and also suppose that there is a periodic point $\hat{x}$ with $\hat{x}_{i^0} = 1$. We want to show that if $\hat{y} = f(\hat{x})$, then *successor* will return a green or uncolored coordinate j with $\hat{y}_j = 1$. Note that dist$(f(0), \hat{y}) =$ dist$(0, \hat{x})$ by Lemma 4.1, so if x moves from 0 to $\hat{x}$ along a shortest path, then $y = f(x)$ must move from $f(0)$ to $\hat{y}$ along a shortest path. This is precisely what happens when successor (i^0) is called. The initial assignment $x \leftarrow x \oplus e_i$ with $x = 0$ and $i = i^0$ clearly moves x from 0 towards $\hat{x}$. Suppose that up to right after some execution of $x \leftarrow x \oplus e_i$, the points x and y have moved along shortest paths from 0 to $\hat{x}$ and from $f(0)$ to $\hat{y}$, respectively. Since the last assignment $x \leftarrow x \oplus e_i$ moved x towards $\hat{x}$, we have dist$(x, \hat{x}) <$ dist$(y', \hat{y})$, so by Lemma 2.1, the assignment $y \leftarrow f(x)$ must move y towards $\hat{y}$ along some dimension j. Then $y = y' \oplus e_j$ for some coordinate j, this coordinate j is non-black by invariant (1), since $\hat{y}_j = 1$, and j is unmarked because each dimension is traversed at most once in a shortest path. If j is either green or uncolored, then *successor* returns j with $\hat{y}_j = 1$ as claimed. If j is red and pred$[j] = i'$, then $\hat{x}_{i'} = \hat{y}_j = 1$ by invariant (3). Furthermore, coordinate i' is different from all previously encountered i because i' is red and the only red$j$ such that pred$[j] = i'$ was unmarked up to now. So the next execution of $x \leftarrow x \oplus e_i$ with $i = i'$ sets $x_{i'} = 1$ and moves x towards $\hat{x}$. Hence our shortest path assumption for x and y holds through one more iteration. Since there is only a finite supply of unmarked red j, eventually some j which is either green or uncolored and satisfies $\hat{y}_j = 1$ is obtained and returned, as claimed.

This fact about *successor* is all we need to know about this procedure, and we use it to show that the three invariants are preserved. The coordinates are partitioned at all times into three types: black coordinates, cycles of red coordinates, where pred$[j]$ always indicates the red coordinate preceding j in the cycle, and a single path of green coordinates, where i is the last coordinate on the path, pred$[j]$ always indicates the coordinate preceding j on the path, and pred$[j]$ is undefined for the first coordinate on the path. The input i to *successor* is always green. If the output j is uncolored, then setting pred$[j] = i$ and coloring j green preserves invariant (2), because *successor* guarantees that if $x_i = 1$ then $y_j = 1$ for x and $y = f(x)$ periodic. If the output j is green, then again setting pred$[j] = i$ preserves invariant (2) for the same reason, and also completes a cycle of green coordinates. Interchanging pred$[j]$ and i, in fact, separates this cycle from the path, leaving a shorter path whose last coordinate i is the coordinate that used to precede j on the path before the cycle was completed. (In the special case where j was the first coordinate on the path, the completed cycle covers the entire path, and a new path is strated somewhere else by picking some uncolored i and coloring i green.) We must show that coloring all coordinates in the cycle red preserves invariant (3). Let i be a coordinate on this cycle, let x be a periodic point, let i' be the coordinate that is obtained by moving forward in the cycle, let x be a periodic point, let i' be the coor-

dinate that is obtained by poving forward in the cycle (folowing the pred pointers backwards) l times, starting at i, and let $x^l = f^{(l)}(x)$. Then $x^l_{i^l} \leqslant x^{l+1}_{i^{l+1}}$ by invariant (2), because $\text{pred}[i^{l+1}] = i^l$. But $x^{M!} = x^0$ and $i^{M!} = i^0$ (since periods and cycle lengths are at most $M = 2^m$), so we must in fact have $x^l_{i^l} = x^{l+1}_{i^{l+1}}$ for all $0 \leqslant l < M!$. In particular, if $j = i^1$ and $y = x^1$, then invariant (3) holds for i, j, x, and y. We must finally show that invariant (1) is preserved when all coordinates in the green path are colored black. Invariant (1) holds for the last coordinate i on the path, because *successor* failed for this last coordinate and we know that *successor* does not fail if there exists a periodic point x with $x_i = 1$. If invariant (1) holds for some coordinate j on the path, then it must also hold for $i = \text{pred}[j]$, because all periodic points x and $y = f(x)$ satisfy $y_j = 0$ by invariant (1) and $x_i \leqslant y_j$ by invariant (2), so that $x_i = 0$. Therefore invariant (1) holds for all coordinates on the green path when the path is re-colored black. ∎

The algorithm also gives the 2-SAT clauses that we need to chaacterize the fixed points of f. Since every periodic point satisfies $f(x) = g(x)$, we can break the fixed point condition $f(x) = x$ into two separate conditions $f(x) = g(x)$ and $g(x) = x$. The first condition is guaranteed by clauses on the black and on the red coordinates. If i is a black coordinate, then we include a clause that foces $x_i = 0$. Let k and k' be red coordinates. We say that k' is *implied* by k if the variable i takes the value k' during some call to successor(k) in the execution of the algorithm. We include a clause $\overline{x_k} \vee x_{k'}$ whenever k' is implied by k, and a clause $x_k \vee x_{k'}$ whenever the m-bit vector x with a 1 in each coordinate implied by k or by k' (or by both) and a 0 everywhere else does not satisfy $f(x) = g(x)$. The remaining condition $g(x) = x$ is guaranteed by clauses that force $x_{\sigma(j)} = x_j$ for all red coordinates j.

LEMMA 6.2. *The 2-SAT clauses mentioned above characterize the set of all fixed points of f.*

Proof. The set S of all points x that satisfy $f(x) = g(x)$ is a median set, because S is the set of fixed points of $g^{-1} \circ f$. By Lemma 4.2, there is a 2-SAT instance I that characterizes the set S. We first claim that if $\hat{x}$ is in S and i is black, then $\hat{x}_i = 0$. If this were not the case, consider the black coordinate i^0 such that $\hat{x}_{i^0} = 1$ that was colored black earliest during the algorithm. Consider the last call to successor(i^0) during the execution of the algorithm. If $\hat{y} = f(\hat{x}) = g(\hat{x})$, then $\text{dist}(0, \hat{x}) = \text{dist}(f(0), \hat{y})$, because g preserves distances. So if x moves from 0 to $\hat{x}$ along a shortest path, then $y = f(x)$ will move from $f(0)$ to $\hat{y}$ along a shortest path. This is what happens during the execution of *successor*, because x_i^0 is set to 1 the first time x is changed, moving x closer to $\hat{x}$, and whenever y is moved closer to $\hat{y}$ by setting $y_j = 1$ for some j such that $\hat{y}_j = 1$, the following assignment to x sets $x_i = 1$ for $i = \sigma(j)$, and, since $\hat{x}_{\sigma(j)} = \hat{y}_j = 1$, this assignment moves x in turn closer to $\hat{x}$. Therefore the call to *successor* can only fail if y_j is se to 1 for some black j. But then $\hat{x}_j = \hat{y}_j = 1$ and j was colored black before i^0 was, acontradiction. On the other hand, if *successor* does not fail and outputs some j with $\hat{y}_j = 1$ then i^0 will be eventually colored black by the main algorithm only if j is colored black before

i^0 is colored black, and we have as before $\hat{x}_j = \hat{y}_j = 1$ with j colored black before i^0, a contradiction.

We can therefore include clauses $\hat{x}_i = 0$ in I for all black coordinates i, and then simplify I so that the remaining clauses only mention variables that correspond to red coordinates. There can only be in I two types of clauses involving two variables x_k and $x_{k'}$ with k and k' red, namely clauses of the form $\overline{x_k} \vee x_{k'}$ and clauses of the form $\overline{x_k} \vee \overline{x_{k'}}$; no clause can be of the form $x_k \vee x_{k'}$ since the all-zero bit vector is in S. We find all such clauses. Suppose that k' is implied by k, and that there is a point $\hat{x}$ in S with $\hat{x}_k = 1$. As we argued in the previous paragraph, a call to successor(k) will only set x_i to 1 if this assignment moves x closer to $\hat{x}$, i.e., if $\hat{x}_i = 1$. Since k' is implied by k, some call to successor(k) sets $x_{k'}$ to 1, and so $\hat{x}_{k'} = 1$. This shows that $\hat{x}_k \leqslant \hat{x}_{k'}$ and justifies all the clauses of the form $\overline{x_k} \vee x_{k'}$ where k' is implied by k.

Note that if k' is implied by k, then in fact $x_{k'}$ is set to 1 during the *last* call to successor(k). This is so because every execution of successor(k) is an extension of the previous execution of successor(k): Once a coordinate j becomes red or black, it will remain the same color until the end of the algorithm and with pred[j] unchanged. Therefore the point x^k with a 1 in each coordinate implied by k and a 0 everywhere else is precisely the *last* value taken by x during the *last* call to successor(k) for this k. The corresponding bits set to 1 in $y = f(x)$ during this execution of *successor* are precisely the bits j for which bit $i = \sigma(j)$ is set to 1 in x (in particular, the last such j will have $\sigma(j) = k$ once the red cycle containing k is closed). Therefore the final values of x and $y = f(x)$ after the last call to successor satisfy $y_j = x_{\sigma(j)}$ for all j, or $y = g(x)$. Thus $f(x^k) = g(x^k)$ and x^k is in S. So there cannot be clauses in I of the form $\overline{x_k} \vee x_{k'}$, where k' is not implied by k: the point x^k is in S but would violate such a clause. This shows that we already have *all* the clauses of the form $\overline{x_k} \vee x_{k'}$ with k and k' red.

Finally, if some point x in S has $x_k = x_{k'} = 1$, then x must necessarily have $x_i = 1$ if i is implied by k or by k'. The median $x^{kk'} = \text{med}(x, x^k, x^{k'})$ is then in S, and $x_i^{kk'} = 1$ if and only if i is implied by k or by k' (just take bitwise majority). So we can decide whether we can include in I a clause $\overline{x_k} \vee \overline{x_{k'}}$, i.e., whether some point x in S has $x_k = x_{k'} = 1$, by testing just one such x, namely the point $x^{kk'}$ with $x_i^{kk'} = 1$ iff i is implied by k orby k', for membership in S.

This completes the characterization of S, i.e., of the points x such that $f(x) = g(x)$. The points x such that $g(x) = x$ are precisely the points with $x_{\sigma(j)} = x_j$, by definition, and we need only consider j red because $\sigma(j) = j$ for j black. The condition is enforced by clauses $\overline{x_{\sigma(j)}} \vee x_j$ (the converse $\overline{x_j} \vee x_{\sigma(j)}$ is redundant). ∎

THEOREM 6.1. *In a network of gatewidth c, an instance of 2-SAT with $O(cm)$ clauses that characterizes all the stable configurations can be found in $O(c^2 m)$ time. All the clauses are local, involving either two inputs or an input and an output to the same gate.*

Proof. The key observation to bound both the running time and the number of clauses is that if pred[j] = i, then i and j are an input and an output, respectively,

to the *same* gate h (ignore the special case where we set $\text{pred}[j]=j$ when j is colored black). This invariant is preserved inductively because successor(i^0) operates entirely within the gate h one of whose inputs in i^0, i.e., during the execution of *successor* the coordinate j can only be an output to h and the coordinate i can only be an input to h. So *successor* returns an output j of h, and setting $\text{pred}[j]=i^0$ preserves the invariant. Therefore each call to *successor* runs in time $O(c)$ (either i or j can only take c different values). After each call to *successor*, some i is either colored or re-colored. The number of such colorations is $2m$, because each coordinate is colored twice, green the first time, and either red or black the second time. The running time is thus $O(cm)$. All the clauses but those of the form $\overline{x_k} \vee \overline{x_{k'}}$ with k, k' red are found directly during the execution of the algorithm in $O(cm)$ time, and so there are at most $O(cm)$ of them. The clauses $\overline{x_{\sigma(j)}} \vee x_j$ involve an input and an output to the same gate, and the clauses $\overline{x_k} \vee x_{k'}$, where k' is implied by k, involve two inputs to the same gate. The clauses $\overline{x_k} \vee \overline{x_{k'}}$ must also involve two red inputs to the same gate. If k and k' are inputs to two different gates h and h', then the point x^k with a 1 in each oordinate implied by k and 0 elsewhere only has 1s in inputs to h, and the corresponding $x^{k'}$ only has 1s in inputs to h'. The corresponding $f(x^k)=g(x^k)$ and $f(x^{k'})=g(x^{k'})$ will have 1s at the outputs of h and h', and if $x^{kk'}$ has a 1 where either x^k has a 1 or $x^{k'}$ has a 1, then $f(x^{kk'})$ has a 1, where either $f(x^k)$ or $f(x^{k'})$ has a 1, so that $f(x^{kk'})=g(x^{kk'})$ and the clause $\overline{x_k} \vee \overline{x_{k'}}$ is not included. So such a clause can only be included if k, k' are red inputs to the same gate. For every choice of a red k, the number of red inputs k' to the same gate that k is an input to is at most c (since pred pairs-up red outputs and red inputs to the same gate), so the number of clauses of this type is also $O(cm)$. The time to test each of them is at most $O(c)$, because only the $2c$ coordinates implied by k or by k' must be set to 1, bounding the total running time by $O(c^2m)$. ∎

One can prove that the algorithm is correct even if no fixed point exists. In that case the starting point 0 is chosen to be any periodic point x such that $\text{dist}(x, f(x))$ is minimum. The algorithm will still produce an isomorphism $g(x_1 \cdots x_m) = f(0) \oplus x_{\sigma(1)} \cdots x_{\sigma(m)}$, such that $f(x)=g(x)$ for all periodic points, and characterize the points x such that $f(x)=g(x)$. Then, although there may be no fixed points, the algorithm can characterize all the *minimal fixed cubes* Q. The dimensions i contained in Q (i.e., those i such that Q contains points x with $x_i=0$ and with $x_i=1$) are just those i such that $y_j=1$ for some j belonging to the same cycle of σ as i, where $y=f(0)$. If we remove all occurrences of such variables in the 2-SAT instance for fixed points described above, we obtain a 2-SAT instance on the remaining variables that characterizes the minimal fixed cubes.

Both in the case of fixed points and in the more general case of minimal fixed cubes, the output of the algorithm is independent of the choices made by the algorithm, i.e., the isomorphism g and the 2-SAT clauses are the same regardless of how a valid starting point 0 is chosen and of the order in which uncolored coordinates are picked by the algorithm. This is not a priori obvious since there can be more

than one isomorphism g that coincides with f in the set of periodic points (although two such g can only differ in the trivial coordinates of the attractor set of f). The mapping g obtained by the algorithm will not be, in general, the same mapping that was used in the proof of Theorem 4.1, and we do not have a natural combinatorial interpretation for the behaviour of this g outside the attractor set (in the spirit of Theorem 4.1).

The 2-SAT instance for the set of points that satisfy $f(x) = g(x)$ is acyclic and transitively closed (see Section 7). The same cannot be said about the 2-SAT instance for the fixed points, whose strong components (see Section 7) correspond to the cycles of σ and where taking the transitive closure could increase the number of clauses from $O(m)$ to $O(m^2)$. The situation is in a sense worse for periodic points. A perodic point x must not only satisfy $f(x) = g(x)$, but also $f(x^r) = g(x^r)$ for $x^r = g^{(r)}(x)$ and $r \geqslant 0$, so we must replace the clauses listed above involving two variables x_k and $x_{k'}$ by corresponding clauses involving $x_{\sigma^{(r)}(k)}$ and $x_{\sigma^{(r)}(k')}$ with $0 \leqslant r \leqslant m^2$. This can increase the number of clauses quadratically, even if we do not take the transitive closure. (A function requiring $\Omega(m^2)$ clauses to decribe its attractor set is, for instance, the one in the example after the proof of Theorem 4.4.) It is therefore a fortunate fact that, at least when the transitive closure is not taken, the fixed point set only requires $O(m)$ clauses. Much of the material in the next section concerns algorithms that gain efficiency by not computing this transitive closure explicitly.

The result in this section allows us to efficienctly reduce questions of enumeration and optimization for adjacency-preserving networks to the corresponding question about 2-SAT instances. Our algorithmic results for 2-SAT appear in a separate paper [12]; the next section gives a brief summary.

7. Algorithms for 2-SAT

A 2-SAT instance on the variables $x_1, ..., x_m$ can be viewed as a directed graph G, the *implication graph* of the 2-SAT instance, whose vertices are the literals $x_1, \overline{x_1}, ..., x_m, \overline{x_m}$, and where the clauses $u \vee v$ are viewed as two implications $\bar{u} \to v$ and $\bar{v} \to u$, represented by directed edges from $\bar{u}$ to v and from $\bar{v}$ to u in G. A 2-SAT instance is then said to be *transitively closed* (resp. *acyclic*) if its implication graph is transitively closed (resp. acyclic), and the transitive closure of a 2-SAT instance is the 2-SAT instance corresponding to the transitive closure of the implication graph. If there is a path from u to v and from v to u in the 2-SAT instance, then $u = v$ for all solutions and we can treat u and v as the same literal. One can identify all such u and v by running a strong components algorithm (in $O(m)$ time [47]) on the directed graph. If the literals in the same strong component are now replaced by a single literal, then the implication graph becomes acyclic. An acyclic 2-SAT instance cannot have equivalent variables. One of the many advantages of this simplification is that the set of solutions is then a connected set (Lemma 4.2), and we can move from each solution to another solution by changing just one bit x_i.

We use several parameters in our algorithms. The number of clauses plus variables in the 2-SAT instance is m. The *degree* d of an acyclic 2-SAT instance is the maximum outdegree of a vertex in the corresponding acyclic implication graph. The degree for a general 2-SAT instance is the degree of the corresponding acyclic 2-SAT instance (with strong components merged). The *width* w of a 2-SAT instance is the cardinality of a largest set of literals S such that for every pair of distinct literals u and v in S, the implication graph contains no directed path from u to v. A *path cover* for a 2-SAT instance is a set of disjoint directed paths in the transitive closure of the implication graph G, whose union covers all the vertices of G (all the literals of the 2-SAT instance). The *width* of a path cover is the number of paths in the path cover.

The efficiency of some of the algorithms depends on the existence of a path cover of small width. By Dilworth's theorem [9], the minimum width of a path cover for a 2-SAT instance is just the width of the 2-SAT instance. If no path cover of small width is known, one can always find a path cover of minimum width by running an expensive min-flow algorithm. Alternatively, if the width of a 2-SAT instance is w, then a greedy algorithm finds a path cover of width $O(w \log m)$ in $O(wm \log m)$ time. Note that if a path cover of width w is known, then the degree d of the 2-SAT instance can be reduced to satisfy $d \leqslant w$ in $O(m)$ time. For if a literal u has outdegree $w+1$, then there must exist two literals v and v' such that $u \to v$ and $u \to v'$ are edges in the implication graph, and v, v' are in the same path with v preceding v'; in that case the edge $u \to v'$ can be removed without affecting the transitive closure and hence the set of solutions of the 2-SAT instance.

THEOREM 7.1. *The solutions of an acyclic 2-SAT instance I can be enumerated with $O(m)$ preprocessing time in $O(d)$ on-line time per solution and using $O(m)$ space, where m is the number of clauses and d is the degree of I. Each solution listed differs from the previous one listed in at most three coordinates.*

The *compatibility graph* of a 2-SAT instance is an undirected graph on the set of literals with an edge (u, v) for each clause $u \vee v$. We assume that the edges $(u, \bar{u})$ are always present. A partial solution to a 2-SAT instance is an assignment of values to a subset of the variables which can be extended to a complete solution by assigning values of the remaining variables. A partial assignment can be represented by a subset S of the vertices in the compatibility graph: if u has been assigned the value 1 then put the vertex uu in S, if u has been assigned the value 0 then put the vertex $\bar{u}$ in S, and if u has not been assigned a value then put both u and $\bar{u}$ in S. A fruitful fact which is (either implicitly or explicitly) present in many 2-SAT algorithms is the following. Recall that a vertex cover for an undirected graph is a subset S of its vertices with the property that at least one of the two endpoints of every edge is in the subset S.

LEMMA 7.1. *The partial solutions to a transitively closed 2-SAT instance are the vertex covers of its compatibility graph; the solutions to a transitively closed 2-SAT*

instance are the minimal vertex covers of its compatibility graph (assuming that a solution exists).

Proof. Transitive closure in the implication graph corresponds to closure under the *resolution* rule in the compatibility graph. That is, if there is a clause $u \vee v$ and a clause $v \vee w$, then we can infer the clause $u \vee w$.

A solution to the 2-SAT instance is a vertex cover of the compatibility graph because at least one literal of each clause is satisfied. It is minimal vertex cover because only one of $u, \bar{u}$ is in the vertex cover and the edge $(u, \bar{u})$ must be covered. To prove the converse, suppose that we have a minimal vertex cover. Suppose also that for some literal v, both v and $\bar{v}$ are in this cover. Consider first the case where there is no self-loop edge (v, v) or $(\bar{v}, \bar{v})$. By minimality, v cannot be removed from the cover, so there must exist an edge (u, v) with u not in the cover. Similarly, $\bar{v}$ cannot be removed from the cover, so there is an edge (v, w) with w not in the cover. But then, by resolution, there is an uncovered edge (u, w), a contradiction. If there is one self-loop, say the edge $(\bar{v}, \bar{v})$, but not the edge (v, v), then we can still conclude that there is an edge (u, v) with u not in the cover, but then resolution yields edges $(u, \bar{v})$ and (u, u), so u must be in the cover, a contradiction. If both self-loops (u, u) and $(\bar{u}, \bar{u})$ are present, then the 2-SAT instance has no solution. Therefore, if a solution exists, then every minimal vertex cover contains only one of $u, \bar{u}$ for each such pair, and hence defines a valid 2-SAT solution. Partial solutions correspond to vertex covers because every vertex cover can be reduced to a minimal vertex cover. ∎

The correspondence between 2-SAT and vertex cover also works in the opposite direction. Given an undirected graph G, we can make G the compatibility graph of a transitively closed 2-SAT instance by associating a positive (unnegated) literal u with each vertex of G and adding dangling edges $(u, \bar{u})$. The vertex covers of G correspond then to the minimal vertex covers of the compatibility graph (just add $\bar{u}$ to the vertex cover if u is not in the vertex cover of G).

This link to vertex cover gives an approach to optimization problems for 2-SAT. The weighted minimization problem assigns a nonnegative weight to each literal in a 2-SAT instance and asks for a solution to 2-SAT that minimizes the sum of the weights of the satisfied literals. By the above correspondence, this is a minimum weight vertex cover problem, which is $\mathcal{NP}$-complete even if all weights are 0 or 1, but has a factor-of-two approximation algorithm even for arbitrary nonnegative weights [38, 30, 4, 7, 26]. Any improvement over this factor of two in 2-SAT would give an improvement in vertex cover and vice versa. The reduction from minimum vertex cover works by giving a weight of 1 to the positive literals and 0 to the dangling negative literals in the compatibility graph constructed from G above; if we give instead a weight of 0 to the positive literals and 1 to the dangling negative literals, and ask for a solution of *maximum* weight, we obtain a reduction from the maximum independent set problem, which is not only $\mathcal{NP}$-complete but very hard to approximate.

An important special case arises when the 2-SAT instance is *bipartite*, i.e., when the compatibility graph is bipartite. A 2-SAT instance becomes bipartite when it has two known solutions a^0 and a^1 such that every solution x lies in between these two (i.e., $\text{med}(a^0, a^1, x) = x$). If this is the case, the trivial variables are the variables x_i such that $a_i^0 = a_i^1$, and once these trivial variables have been discarded so that $a_i^0 \neq a_i^1$, all remaining clauses must be of the form $x_i = a_i^0 \vee x_j = a_j^1$ (otherwise either a^0 or a^1 will not be a solution), so the compatibility graph is bipartite. Conversely, a bipartite 2-SAT instance always has two solutions a^0 and a^1 with the above property, namely the two solutions obtained by setting the literals on one side of the bipartite graph to 0 and those on the other side to 1. From the point of view of network stability, bipartite instances arise hen the adjacency-preserving network is monotone (i.e., $\text{med}(x, y, 0) = y$ implies $\text{med}(f(x), f(y), 0) = f(y)$). Such networks have a zero-most fixed point and a one-most fixed point both of which can be found in $O(m)$ time [36], providing us with the two points a^0 and a^1.

For bipartite graphs, the minimum weight vertex cover problem can be solved in polynomial time by bipartite matching if all weights are 0 or 1, or by *uncapacitated* maximum flow in the case of arbitrary weights. This gives a polynomial-time algorithm for the weighted bipartite 2-SAT problem. The main difficulty in applying matching algorithms to the unweighted 2-SAT problem is that the 2-SAT instance is usually not transitively closed, and preprocessing the 2-SAT instance by taking a transitive closure would be time-consuming and could also considerably increase the number of edges. Our algorithm handles this difficulty by only taking transitive closures *implicitly*. The running time is $O(\sqrt{K}m)$ when the weight K of the optimal solution is small ($K = O((m/\log^2 m)^2)$. We thus get the same running time as in matching algorithms, with no penalty for the implicit transitive closure. Flow algorithms can beapplied directly, without taking a transitive closure. The difficulty here is that the usual bounds for flow involve the number of vertices in the graph, and in our intended application the number of vertices could be large (proportional to the number of edges m). Only the *width* of the 2-SAT instance is small. Our algorithm runs in $O(wm \log K)$ time for arbitrary K, thus matching the running time of known flow algorithms up to a scaling factor of $\log K$ but with the number of vertices replaced with the possibly much smaller width w (we assume that a path cover of width w is given). We do not know whether our algorithm can be imporved by replacing the logarithm with a double logarithm as in [1] or by eliminating the dependency on K to obtain a strongly polynomial algorithm (without increasing the main term to m^2).

THEOREM 7.2. *Given a bipartite 2-SAT instance with m clauses and a path cover of width w, a minimum-weight solution can be found in* $O(\min(w, \sqrt{K})(m \log(K/w^2 + 2) + \min(w \log w, \sqrt{K} \log^2 w))$ *time, where K is the weight of an optimal solution.*

Both in our application to stable marriage and in the above algorithm, the weights on the literals are not given individually, but rather as partial sums of the weights in each path of the path cover, where each sum ranges over an initial

segment of the path. The running time remains as stated even if the stronger logarithmic (bit) model of computation is adopted.

The algorithm finds more than just one solution of minimum weight: it outputs a bipartite 2-SAT instance whose solutions are all the minimum weight solutions. In fact this 2-SAT instance is the same as the original instance with the addition of clauses that force equality between certain literals (i.e., equivalent variables have been created). This can be useful if we need to run a tie-breaking algorithm on the set of optimal solutions.

The known approximation algorithms for vertex cover can be viewed as blocking flow computations. This blocking flow interpretation has the advantage of being valid even before the 2-SAT instance has been transitively closed. We achieve the best known running time for blocking flows [17] with the number of vertices replaced by the possibly smaller width w of a known path cover (but without the generality of capacitated blocking flows).

THEOREM 7.3. *A solution within a factor of* 2 *of the optimum for the minimum weight* 2*-SAT problem can be found in* $O(m \log(w^2/m+2))$ *time for* 2*-SAT instances with* m *clauses with a known path cover of width* w.

Another application of the width of a 2-SAT instance arises when we want to determine the trivial variables of a 2-SAT instance. This can easily be done in $O(m)$ time for bipartite instances as we saw before. In fact even if the 2-SAT instance is only implicitly bipartite in the sense that the two solutions a^0 and a^1 mentioned above exist but are not known, one can find these two olutions (and hence the trivial variables) in $O(m\alpha(m))$ time, where $\alpha(m)$ is the inverse Ackerman function. The complexity in the case of nonbipartite instances is greater, and we know of no way of obtaining the trivial variables other that by computing the transitive closure.

LEMMA 7.2. *All the clauses in the transitive closure of a* 2*-SAT instance involving at least one literal from a fixed path* P *can be obtained in* $O(m)$ *time. Thus the transitive closure can be obtained in* $O(wm)$ *time if a path cover of width* w *is known.*

Proof. Let $u_1 \to u_2 \to \cdots \to u_r$ be the path P. For every literal v, let $\phi(v)$ be the least i such that the clause $u_i \vee v$ is in the transitive closure of the 2-SAT instance. If no such i exists, we let $\phi(v) = r+1$. Note that a clause $u_j \vee v$ is in the transitive closure iff $j \geqslant \phi(v)$. To obtain $\phi(v)$ for all literals v, we consider $i = 1, 2, \ldots, r$ in turn. For each i, we find all literals v that are reachable from $\overline{u_i}$ in the implication graph, set $\phi(v) = i$, since $\overline{u_i} \to v$ is in the transitive closure, and remove these literals and incident edges from the implication graph. After the last value $i = r$ has been considered, we set $\phi(v) = r+1$ for all literals v that were not removed from the implication graph. The time complexity is $O(m)$, since every vertex and every edge of the implication graph is considered and removed only once. ∎

As observed in [24], once the transitive closure is known, and the existence of

at least one solution is known (this can be determined in $O(m)$ time), one can check whether an assignment of values to k variables is a partial solution in $O(k^2)$ time. For if the k literals $u_1, u_2, ..., u_k$ have been set to 0 in this partial assignment, then there can be no clause $u_i \vee u_j$ in the transitive closure if this partial assignment is a partial solution. Conversely, if there is no clause $u_i \vee u_j$ in the transitive closure, then the literals $\overline{u_1}, \overline{u_2}, ..., \overline{u_k}$ added to the literals $v, \bar{v}$ such that neither v nor $\bar{v}$ is one of the u_i form a vertex cover of the compatibility graph of the transitive closure, and hence a partial solution by Lemma 7.1.

In particular, one can check in constant time whether a variable x_i is a trivial variable by testing the assignments $x_i = 0$ and $x_i = 1$.

We conclude this section on 2-SAT with the following parallel complexity fact:

LEMMA 7.3. *(Acyclic) transitively closed 2-SAT instances can be solved in* $\mathcal{NC}^1$.

The proof is a slight modification of the Cook and Luby [8] algorithm. Set $x_i = 1$ if the outdegree of x_i in the implication graph of the 2-SAT instance is smaller than the outdegree of $\overline{x_i}$, and set $x_i = 0$ otherwise. The correctness of this algorithm follows from the fact that for each clause $u \vee v$ we have outdegree$(\bar{u}) >$ outdegree(v) and outdegree$(\bar{v}) >$ outdegree(u), so either outdegree$(\bar{u}) >$ outdegree(u) or outdegree$(\bar{v}) >$ outdegree(v), and the clause $u \vee v$ is satisfied by the above assignment. The acyclicity condition can be removed by ignoring variables that are equivalent to lower-indexed variables.

8. STABLE MATCHING PROBLEMS

Subramanian [45] discovered an elegant reduction from stable roommates to network stability problems over X-gates. An X-gate is a two-input, two-output adjacency-preserving gate with inputs u, v and outputs $u\bar{v}, v\bar{u}$. For each person i is a stable roommates instance, introduce boolean variables x_{ij} for all $0 \leqslant j \leqslant k_i$ is the length of the preference list of person i. The interpretation of these boolean variables is that x_{ij} is true in a given matching if and only if person i is not paired up with one of his top j choices in that matching. The key observation is that the stable matchings can be characterized by two sets of conditions on these boolean variables, namely (a) $x_{i0} = 1$ for each person i and (b) $x_{ij} = x_{i(j-1)}\overline{x_{i'(j'-1)}}$, $x_{i'j'} = x_{i'(j'-1)}\overline{x_{i(j-1)}}$ if i' is the jth person in the list of person i and i is the jth person in the list of person i'. This is the standard stability condition, which requires that if i and i' are in each other's preference lists, then they must be matched unless one of them is matched to someone he likes better. (If we also wish to require that every person be matched, then we just add conditions $y_i = x_{ik_i} \oplus y_i$ with a dummy variable y_i for each person i. This modification is not required if the instance is complete, i.e., every person has listed every other person as a candidate roommate, or every man has listed every woman and vice versa in a marriage instance.)

The problem then becomes a network stability problem, with edges corresponding to the x_{ij} variables. The input edges are the x_{i0} and are set to 1 to guarantee condition (a); condition (b) is ensured by using X-gates with inputs $x_{i(j-1)}$, $x_{i'(j'-1)}$ and outputs x_{ij}, $x_{i'j'}$.

The structural theorems of Section 4 have here some interesting consequences. The fact that the median of three stable configurations is also a stable configuration tells us that given three stable assignments, if each person picks from the three choices obtained in these assignments the intermediate one (from the point of view of his own preference list), the result will also be a valid assignment and, in fact, a stable one. Another important structural observation is that certain cyclic sequences of pairs of people known as *rotations*, which proved to be central to the study of stable matching problems (see [21, 22, 31–33]), correspond to the disjoint cycles of the permutation σ associated with the isomorphism of Therem 4.1. The new definition arising from this theorem is mathematically appealing, and also turns out to be very useful in the study of stability problems from the point of view of parallel complexity (see Section 9).

We know that the set of stable assignments can be characterized by an instance of 2-SAT on the x_{ij} variables. This 2-SAT instance corresponds to the partial orders of earlier work on stable matching [21, 22, 31–33]. This 2-SAT instance has a path cover of width at most $2n$, because all the variables x_{ij} corresponding to a fixed person i form a path (since $x_{i(j+1)}=1$ implies $x_{ij}=1$), and similarly for the negated literals $\overline{x_{ij}}$. As noted in the last section, this also implies that the degree (after strong components have been collapsed) is also at most $2n$. In the case of the stable marriage problem, Subramanian showed that the network is equivalent to a monotone network: replace each x_{ij}, where i is an man by a new variable $y_{ij}=\overline{x_{ij}}$; we now have X-gates with inputs $u, \bar{v}$ and outputs $u', \overline{v'}$ and this is equivalent to a comparator with inputs u, v and outputs u', v' (see [45]) which is a monotone gate. Thus the 2-SAT instance is bipartite (this also follows from the existence of a man-optimal and a woman-optimal solution). As noted independently by Gusfield [20], the problem of finding the stable pairs (pairs of people who are matched to each other in some stable solution) reduces to the problem of finding the trivial variables of the 2-SAT instance, a problem that is easier in the bipartite marriage case than in the roommate case. This is so because given a candidate solution x, if person i is *not* matched to the jth person in his preference list in x, then $x_{ij}=x_{i(j-1)}$, and person i will be matched to this jth person in some other solution y iff $y_{ij}\neq y_{i(j-1)}$, i.e., iff x_{ij} and $x_{i(j-1)}$ are neither both trivial nor both nontrivial and equivalent. Equivalence is just membership in the same strong component, so the only nontrivial part is testing triviality. In the roommates case, however, we are only able to find the trivial variables by computing the transitive closure of the 2-SAT instance; then the pairing of person i with his jth choice is a stable pair iff the partial assignment $x_{i(j-1)}=1$, $x_{ij}=0$ is a partial solution, and we know from the last section that this can be tested in constant time from the transitive closure.

These remarks, combined with the results of the last two sections (Theorem 6.1, Theorem 7.1, Lemma 7.2), give the following:

THEOREM 8.1. *In a stable roommates problem with n people and m candidate pairs, $m \leqslant \binom{n}{2}$, the set of stable assignments has a 2-SAT description with $O(m)$ clauses and $O(n)$ width. From this description, which can be found in $O(m)$ time, the stable pairs for each person can be obtained in $O(m)$ time for a total $O(nm)$ time, and the stable assignments can be enumerated in $O(n)$ time per assignment, using $O(m)$ space. In the more restricted stable marriage case, the 2-SAT instance obtained is bipartite (and $O(m)$ time suffices to find all stable pairs* [21]).

We can also go in the other direction and construct a stable roommates instance from a given 2-SAT instance:

THEOREM 8.2. *Every 2-SAT instance with n variables and m clauses characterizes the solutions to a small stable roommates instance, with $O(n)$ people and $O(m)$ candidate pairs, and this roommates instance can be found in $O(m)$ time (the number of candidate pairs can be increased to obtain complete preference lists if so desired). If the 2-SAT instance is bipartite, then the roommates instance obtained is, in fact, a stable marriage instance.*

Proof. We assume that each clause contains two distinct variables (if a clause contains just one variable, then either the clause or the variable can be eliminated). We further assume that the 2-SAT instance is acyclic (this can be enforced by a strong components algorithm which keeps only one copy of each set of equivalent variables).

For each variable x_i introduce four people p_i, $\bar{p}_i$, q_i, $\bar{q}_i$. Each of these people lists the two complementary people in his preference list: Person p_i lists $\bar{p}_i$ and $\bar{q}_i$ in that order, person $\bar{q}_i$ lists p_i and q_i in that order, person q_i lists $\bar{q}_i$ and $\bar{p}_i$ in that order, and person $\bar{p}_i$ lists q_i and p_i in that order. Call the two people that each person has listed so far its two *special* people. Now, for every clause $x_i \vee x_j$, person p_i adds p_j to its preference list anywhere in between its two special people, and similarly p_j lists p_i anywhere in between its two special people. For clauses of the two remaining types $\overline{x_i} \vee x_j$ and $\overline{x_i} \vee \overline{x_j}$, the two people who list each other anywhere in between their two special people are $\bar{p}_i$, p_j for the first type, and $\bar{p}_i$, $\bar{p}_j$ for the second type of clause. This completes the contruction. If complete preference lists are desired, every perso adds every unlisted person to its preference list at the *end* of the list.

Note that each person r will be matched at *worst* with its second special person r', because r is the first person in the list of r'. Furthermore, we claim that each person can *only* be matched with one of its two special people. This is clear for q_i and $\bar{q}_i$, because these two people listed non non-special person before their second special person. So if p_i is not matched to one of its two special people then $\bar{q}_i$ will be matched to q_i and $\bar{p}_i$ will not be matched to one of its two special people either; conversely, if $\bar{p}_i$ is not matched to one of its two special people, then $\overline{q_i}$ will be matched to $\bar{q}_i$ and p_i will not be matched to one of its two special people either. So p_i is matched to a non-special person iff $\bar{p}_i$ is matched to a non-special person.

Now consider only the variables x_i, where p_i, $\bar{p}_i$ are matched to non-special people, and only the clauses involving such variables. If we view these clauses as implications, then each literal x_i and $\overline{x_i}$ must be the antecedent of some implication. For example, there must exist some clause of the form $x_i \to u$, i.e., $\overline{x_i} \vee u$, because person $\bar{p}_i$ is matched to a non-special person (since non-special matchings come from clauses in the 2-SAT instance). But this means that each literal has outdegree at least one in the graph defined by these implications, and hence the 2-SAT instance contains a cycle, contradicting the assumption of acyclicity.

Therefore no person can be matched to a non-special person, i.e., each person is matched to one of its two special people. This leaves only two possibilities for the four people indexed by i, i.e., either they pair-up as $(p_i, \bar{p}_i)$, $(q_i, \bar{q}_i)$, or they pair-up as $(p_i, \bar{q}_i)$, $(q_i, \bar{p}_i)$. Set $x_i = 1$ if they pair-up in the first way, and set $x_i = 0$ if they pair-up in the second way. The resulting bit vector x must then satisfy all the clauses in the 2-SAT instance. For example, if there is a clause $x_i \vee x_j$, then the bit vector x cannot have $x_i = x_j = 0$, because this would mean that both p_i and p_j are matched to their second special person, and they would rather be matched to each other than to their second special person. The other types of clauses are handled similarly by referring to person $\bar{p}_i$ or $\bar{p}_j$ if the negated literals $\overline{x_i}$ or $\overline{x_j}$ are involved. So a bit vector x can only describe a stable solution if it satisfies all the 2-SAT clauses. Conversely, if a bit vector x satisfies all the 2-SAT clauses, the x describes a stable solution. For if there is a violation of stability, it cannot involve people listed after the second special person, because every person is matched to his second special person or better. It cannot involve people who are special to each other, because if they are not matched to each other then at least one of them is matched to his *first* special person and has no desire to switch. Finally, the destabilizing pair cannot be a pair that came from one of the clauses, say the pair (p_i, p_j) corresponding to the clause $x_i \vee x_j$, because, since this clause is satisfied by x, at least one of x_i and x_j must be set to 1 in x, which means that at least one of p_i and p_j is matched to his first special person and has no desire to switch. This accounts for all the pairs, so the solutions to the 2-SAT instance correspond to the solutions to the roommate instance, completing the proof.

If the 2-SAT instance is bipartite, then we can assume modulo some renaming that all positive literals are on one side of the bipartite graph and all the negative literals are on the other side; i.e., all the clauses are of the form $\overline{x_i} \vee x_j$, so all the preference pairs listed involve a negative person ($\bar{p}_i$ or $\bar{q}_i$) and a positive person (p_j or q_j), i.e., two people of opposite sexes (if we wish to extend the preference lists to complete preference lists, we only include preference pairs of this type). ∎

This theorem does not describe the complete picture of the relationship between stable matching and 2-SAT. The reason is that although 2-SAT instances with n variables and m clauses map to matching instances with $O(n)$ people and $O(m)$ pairs, one can construct matching instances with n people and m pairs for which the corresponding 2-SAT instance has $O(m)$ clauses but requires $\Omega(m)$ variables (even after equivalent variables have been merged). The more natural relationship seems

to be the one that maps matching instances with n people and m pairs to 2-SAT instances with $O(n)$ *width* and $O(m)$ clauses. Unfortunately, even here, the converse does not hold, i.e., some 2-SAT instances with n width and m clauses have corresponding matching instances with $O(m)$ pairs but which require $\Omega(\sqrt{nm})$ people (and perhaps more). Still, the natural correspondence seems to be between number of people and width. To see this, consider a more general kind of stable roommates problem, in which people are allowed to list each other *more than once*, and each occurrence of person i in the list of person j corresponds to a specific occurrence of person j in the list of person i. This could mean, for example, that i and j have to decide not only whether they will room together, but also who will pay the phone bill, the rent, etc., and that the different options available can make a difference in whether they room together to not. All the algorithms that have been proposed for stable marriage and stable roommates solve, in fact, this more general problem; i.e., none of these algorithms cares about whether two entries in a preference list correspond to the same person or not. With this more general definition of a matching problem, one can prove that matching instances with n people and m pairs correspond to 2-SAT instances of width n with m clauses, in both directions, up to a small constant factor (This more general matching instance also eliminates the need for the $q, \bar{q}$ people in the above construction, since the $p, \bar{p}$ people can be each other's special person twice, with the most preferred occurrence for one of them corresponding to the least preferred occurrence for the other one.) We do not know whether the additional constraint of uniqueness in the preference lists restricts the 2-SAT instances in a way that can be exploited algorithmically.

The main use of this theorem is in proving lower bounds. It gives, for instance, the #P-completeness result for counting stable marriages of Irving and Leather [33], since counting the number of solutions to a bipartite 2-SAT instance is #P-complete [42]. It also gives lower bounds for the "optimal" stable matching problem.

In the "optimal" stable matching problem, the jth entry in the preference list of each person i is assigned a weight $w_{ij} \geq 0$. (If the length of the preference list of person i is k_i, we also include a special weight $w_{i(k+1)}$ when the possibility of i not being paired-up is allowed.) The aim is then to find a stable assignment that pairs up person i with the person in some position $p(i)$ in his preference list (with $p(i) = k+1$ if i is not paired-up) so as to minimize the total weight of the solution, i.e., the sum of the $w_{ip(i)}$ over all i. The *egalitarian* stable matching has $w_{ij} = j$. Note that the weight of a solution x is then simply the number of x_{ij} such that $x_{ij} = 1$. The preceding result, combined with Lemma 7.1, gives the following:

THEOREM 8.3. *The egalitarian stable roommates problem is $\mathcal{NP}$-complete and can be approximated within a factor of α of the optimum if and only if minimum vertex cover can be approximated within the same factor α. The best value known is $\alpha = 2$.*

Proof. An instance of vertex cover on a graph with n vertices and m edges can be viewed as a 2-SAT instance with n variables and m clauses. The clauses are of

the form $x_i \vee x_j$ and correspond to edges (i, j) in the graph, and a variable x_i is set to 1 in a candidate solution iff i is in the candidate vertex cover. A minimum vertex cover this corresponds to a 2-SAT solution x with the fewest x_i set to 1. To express this 2-SAT problem as a stable matching problem, we again do the reduction of the previous theorem, in which two people p_i and p_j list each other between their respective special people iff the clause $x_i \vee x_j$ is present, but with the addition that if p_i lists p_j in this way, then $\bar{p}_i$ lists $\bar{q}_j$ between its two special people as well. This modification does not affect the set of solutions because the pair $(\bar{p}_i, \bar{q}_j)$ cannot be a destabilizing pair, since $\bar{p}_i$ is listed *after* the second special person of $\bar{q}_j$, who is the worst possible mate for $\bar{q}_j$. The only purpose of this modification is to make the *number* of people that p_i and p_i have listed between their two special people the same. Finally, list an extra $\bar{p}_j$ between the two special people of $\bar{p}_i$ (again, this does not affect the stable solutions because $\bar{p}_j$ still lists $\bar{p}_i$ after its second special person). Now the sum of the weights for the four people p_i, $\bar{p}_i$, q_i, $\bar{q}_i$ is a candidate solution x in which the pairing $(p_i, \bar{p}_i)$, $(q_i, \bar{q}_i)$ occurs is *one more* than the sum of their weights if the pairing $(p_i, \bar{q}_i)$, $(q_i, \bar{p}_i)$ occurs. Recall from the proof of the previous theorem that the first pairing corresponds to $x_i = 1$ in the 2-SAT solution and the second pairing to $x_i = 0$. Thus the total weight of a stable matching is one more when $x_i = 1$ than when $x_i = 0$ for each i and is therefore minimized when the number of x_i set to 1 is minimized, i.e., when the vertex cover is a minimum vertex cover. This proves that the egalitarian stable roommates problem is $\mathcal{NP}$-complete.

To obtain the approximation result, we introduce a large number L of extra people, L even, and pair these people up so that each person in a pair lists the other person in the pair *first* in his preference list. We then modify the preceding construction so that instead of listing just one extra person between the two special people of $\bar{p}_i$, we list all L new people between the two special people of $\bar{p}_i$. Thus changing x_i from 0 to 1 will now increase the weight of the stable matching solution by L, and a solution to vertex cover with ρ vertices corresponds to a solution to stable matching of weight $\rho L + \Delta$, where $\Delta = 2m + 6n + L = 2L$ if we choose $L = 2m + 6n$. Now an algorithm with an approximation guarantee of α for the egalitarian matching problem will give a solution of size ρ for the vertex cover problem, where $\rho L + \Delta \leqslant \alpha(\rho^* L + \Delta)$ and ρ^* is the size of a minimum vertex cover. Thus $\rho + 2 \leqslant \alpha(\rho^* + 2)$. To eliminate the "+2," suppose that two vertices in a minimum vertex cover are known. Then after removal of these two vertices and adjacent edges, the minimum vertex cover is of size $\rho^* - 2$, so the above approximation algorithm gives a solution of size ρ', where $\rho' + 2 \leqslant \alpha\rho^*$, and adding the two deleted vertices gives a solution to the original problem of size $\rho = \rho' + 2 \leqslant \alpha\rho^*$, as desired. The two chosen vertices can be picked by trying all possibilities (actually we only need to consider the endpoints of two disjoint edges).

To prove the converse, if an approximation algorithm that guarantees a factor of α for vertex cover is known, we can take an egalitarian roommates instance, find the 2-SAT instance on the x_{ij} variables and the compatibility graph of its transitive closure, and find an approximate minimum vertex cover (which we can assume is also a minimal vertex cover) for the subgraph induced by the positive literals x_{ij}.

The number of literals in such a vertex cover is the same as the number of x_{ij} set to 1 in a solution to the 2-SAT instance (just add the negative literals $\overline{x_{ij}}$ to the vertex cover to obtain a cover of the entire compatibility graph, then discard vertices to obtain a minimal such cover which corresponds to a 2-SAT solution by Lemma 7.1). This solves the egalitarian roommates instance with the same approximation factor. ▮

A natural requirement is to assume that the weights w_{ij} in a person's preference list are either monotonically nondecreasing or monotonically nonincreasing, as a function of j. A less stringent constraint allows the w_{ij} to be a sum $w_{ij}^1 + w_{ij}^2$ of two nonnegative numbers, where the w^1 are monotonically nondecreasing and the w^2 are monotonically nonincreasing. (Every sequence of w_{ij} for a fixed person i can be represented in this way, provided that we first add a sufficiently large constant to the w_{ij} of person i to enture that the two monotonic sequences will be nonnegative. Adding this constant pereserves the optimality or suboptimality of algorithms, but it may affect both the running time and the approximation guarantee of an algorithm.) We shall assume that the w_{ij} are presented in this form.

THEOREM 8.4. *The weighted "optimal" stable roommates problem can be solved within an approximation factor of* 2 *in* $O(m \log(n^2/m))$ *time for instances with n people and m candidate pairs.*

Proof. Assign weight $w_{i(j+1)}^1 - w_{ij}^1 \geqslant 0$ to the literal x_{ij} and weight $w_{ij}^2 - w_{i(j+1)}^2 \geqslant 0$ to the literal $\overline{x_{ij}}$ in the 2-SAT instance for the given "optimal" stable roommates problem (we assume for simplicity that $w_{i0}^1 = w_{i(k+1)}^2 = 0$; i.e., the two monotonic sequences start with 0). In a solution in which person i is matched with his choice j', we have $x_{ij} = 1$ for $j < j'$ and $x_{ij} = 0$ for $j \geqslant j'$, so the sum of the weights of the literals x_{ij} and $\overline{x_{ij}}$ for person i that are set to 1 is precisely $w_{ij'}^1 + w_{ij'}^2 = w_{ij'}$. Hence the minimum weight stable roommate assignment has the same weight as the corresponding minimum weight 2-SAT solution, and the latter problem has by Theorem 7.3 an $O(m \log(n^2/m))$ time approximation algorithm for 2-SAT instances of width $O(n)$ with $O(m)$ clauses. ▮

Note that the above running time is always $O(n^2)$. When the reduction in the preceding theorem is applied to a weighted stable marriages problem, the 2-SAT instance obtained is bipartite and we can obtain an optimal solution using the algorithm from the lst section. An upperbound on the expression from Theorem 7.2 using $m \leqslant (n_2^n)$ gives the following.

THEOREM 8.5. *There is an* $O(\min(n, \sqrt{K})\, m \log(K/m + 2))$ *time algorithm for the weighted "optimal" stable marriage problem, where n is the number of people, m is the number of candidate pairs, and K is the weight of an optimal solution.*

In the egalitarian stable marriage, where all the x_{ij} variables have weight 1, the weight of an optimal solution is $K \leqslant m$, so the theorem gives an $O(m^{1.5})$ algorithm.

Note that the first reduction in Theorem 8.3 also reduces a bipartite vertex cover on graphs with n vertices and m edges to a stable marriage problem with $O(n)$ people and $O(m)$ candidate pairs (for bipartite graphs, the people used in the reduction can be partitioned into two sexes, so the roommates instance is actually a marriage instance). In fact, if the vertices have weights adding up to at most m, then we may introduce n additional pairs of people as in the second reduction of Theorem 8.3 and list w_i of these people in the list of $\bar{p}_i$, where w_i is the weight of vertex i, thus obtaining a reduction from a weighted vertex cover problem with total weight m. (We can assume that $w_i \leqslant n$, because this condition can always be enforced by splitting vertices of weight greater than n, creating at most m/n additional vertices and m additional edges.) By duality, the value of a minimum weight vertex cover in a bipartite graph is the same as the size of a maximum matching in the graph, where the weight of a vertex indicates its multiplicity; i.e., we assume that there are w_i copies of vertex i, and each of them can be matched to any of the copies of vertex j if there is an edge between i and j. This problem is often referred to as the degree-constrained subgraph problem, and the fastest known running time for bipartite graphs with at most n vertices, m edges, and total multiplicity m is the $O(m^{1.5})$ matching bound. Thus an improvement below $O(m^{1.5})$ for the egalitarian marriage problem would imply an improvement in this matching bound. In the general weighted case, the reduction in Theorem 8.3 allows us to reduce the dual of an arbitrary uncapacitated maximum flow problem to a weighted stable matching problem. The fastest known algorithms for this problem run no faster than for the capacitated maximum flow problem, i.e., in $O(nm \log(n^2/m))$ for graphs with n vertices and m edges (see Goldberg and Tarjan [16]) and in $O(nm \log((n/m)\sqrt{\log U + 2}))$ when the largest supply or demand is U (see Ahuja, Orlin, and Tarjan [1]). The Goldberg and Tarjan flow algorithm, when applied to the weightted marriage problem, gives the $O(m^2 \log m)$ time bound of Irving, Leather, and Gusfield [33], because the number of variables in the 2-SAT instance can be as large as m. We do not know whether a strongly polynomial algorithm with the main term equal to nm, where n is the number of people in the marriage instance, can be obtained.

Another application is the lexicographic stable marriage problem, where we wish to maximize the number of people who get their first choice, and within all the maximizing solutions we wish to maximize the number of people who get one of their first two choices, and so on. The problem of maximizing the number of people who obtain one of their first r choices can be stated as a weighted problem by letting $w_{ij} = 0$ for $j \leqslant r$ and $w_{ij} = 1$ for $j > r$, and then K is at most the number n_r of people who have at least r entries in their preference lists. Of course, in the lexicographic problem these subproblems are not independent, but we can solve them one at a time, using at each stage r the original 2-SAT instance plus some extra 2-SAT clauses that characterize *all* the solutions to the problem up to stage $r-1$. The key property is the fact that the optimization algorithm of Theorem 7.2 returns at each stage a characterization of *all* the solutions, not just one solution. The total running time is the sum of at most n terms $O(\sqrt{n_r}m)$. where the n_r add

up to m, which is maximized when $n_r = m/n$ for each r for a total $O(n^{0.5}m^{1.5})$ running time.

The *parametric* egalitarian stable marriage is the egalitarian stable marriage with the prefereces of the men weighted by an additional factor of λ. If $\lambda = p/q$ integer, then we can multiply by a weight of p the preferences of men and by a weight q to the preferences of women, obtaining the same solution. The optimal solution has $K \leqslant m \min(p, q)$. Gusfield and Irving [23] show that we need only consider $1/m \leqslant \lambda \leqslant m$ (the optimal solution is the woman-optimal solution below $1/m$ and the man-optimal solution above m), and that this range of values can be partitioned into intervals $[\lambda_1, \lambda_2]$ such that a solution for λ in the interior of such an interval is optimal in the entire interval. The values λ_1, λ_2 are rational numbers with denominators of at most m. Therefore, $|\lambda_1 - \lambda_2| < 1/m^2$. Either by rounding up or by rounding down λ by less than $1/(2m^2)$, we can obtain a value of $\lambda = p/q$ that gives the same optimal solutions, with $q = 2m^2$. This gives $K \leqslant 2m^3$ and implies an $O(nm \log m)$ bound for the parametric egalitarian stable marriage. Gusfield and Irving [2]3] find solutions for the parametric optimal stable marriage in *all* the intervals $[\lambda_1, \lambda_2]$ in $O(m^2 \log m)$ total time, the same time that their algorithm takes to find a solution for a single λ. An interesting question is whether our algorithm can be modified to find solutions efficiently in all intervals as well.

9. Parallel Complexity

In this section we show that several questions for functions f that are described by adjacency-preserving circuits over a fixed set of gates Ω have the same parallel complexity, up to $\mathcal{NC}^1$ reductions. We prove that finding a fixed point for a function over Ω is no harder than deciding whether a function over Ω has a fixed point, and that this question is in turn as easy as testing whether a given point is a fixed point for a function over Ω (or, equivalently, as easy as evaluating a circuit over Ω). The first result says that the problem of adjacency-preserving network stability is *self-reducible*; i.e., the search problem reduces to the decision problem. This phenomenon is common in polynomial-time complexity, but less so in parallel complexity (see [34] for a discussion of this issue). The second result says that the "deterministic" problem of finding and checking an answer in no harder than the "nondeterministic" problem where we are allowed to guess the right answer before checking it. We view these results as providing further evidence to the thesis [36] that circuit value and network stability problems over sets of adjacency-preserving gates define new, "robust" parallel complexity classes within $\mathcal{P}$.

Theorem 9.1. *Given a set of adjacency-preserving gates Ω containing the negation gate, the following problems are equivalent under log-space uniform $\mathcal{NC}^1$ reductions*:

(a) *Evaluating a circuit over Ω;*

(b) *Deciding whether a network over Ω has a stable configuration.*

(c) *Constructing a stable configuration for a network over* Ω.

(d) *Constructing a description of the set of all stable configurations for a network over* Ω.

(e) *Constructing a description of the set of periodic points and of the behaviour of the network on the set of periodic points for a network over* Ω.

Proof. Problem (a) is no harder than any of the remaining problems, as shown in [36], because a circuit value problem can be stated as a network stability problem by adding a single gate after the output bit u. (This gate is a two-input, one-output gate that enforces either $u \oplus v = v$ or $\neg(u \wedge v) = v$, depending on the type of gates available in Ω; in either case, there exists a fixed point iff $u = 0$; and one of these two types of gate, exclusive-or and nand, must be available except in the easy case where Ω contains just a negation gate.)

We reduce the remaining four problems to problem (a). Given a function f defined by an adjacency-preserving circuit over Ω, $f^{(k)}(x)$ is a periodic point for $k \geqslant m^3$ and x arbitrary, by Theorem 4.4; thus finding a periodic point reduces to circuit evaluation. Even finding a complete description of the set of periodic points is a circuit evaluation problem. For suppose that we wish to determine whether the 2-SAT description for the set of periodic points contains the clause $x_i \vee x_j$. This will be the case iff f contains no periodic point x with $x_i = x_j = 0$. Suppose that such a point x^0 exists, and that in the permutation σ associated with the isomorphism of Theorem 4.1, coordinate i belongs to a cycle of length l_i and coordinate j to a cycle of length l_j. Let l_{ij} be a multiple of $2l_i$ and $2l_j$ greater than $2m^2$. Let y^0 be an arbitrary periodic point (obtained by evaluating $f^{(m^3)}$ at some arbitrary point), and let $y^{t+1} = f^{(l_{ij})}(h_{ij}(y^t))$, where h_{ij} simply replaces the ith and the jth coordinates of its input by 0. Note that y^m is just the output of some Ω-circuit; we claim that y^m is a periodic point with $y_i^m = y_j^m = 0$. To see this, consider the periodic points defined by $x^{l+1} = f^{(l_{ij})}(x^t)$. Each of these periodic points has $x_i = x_j = 0$, because the first one x^0 does and because the values at coordinates i and j recur with periods $2l_i$ and $2l_j$, respectively. Thus $\text{dist}(y^{t+1}, x^{t+1}) \leqslant \text{dist}(y^t, x^t)$, and the inequality is in fact strict if some coordinate of y^t is changed when h_{ij} is applied. Since this distance cannot decrease more than m times, we must eventually obtain a y^t with $y_i^t = y_j^t = 0$. Note that all y^t are periodic points, because if y^t is periodic then $h(y^t)$ is a distance at most 2 from the set of periodic points, and this forces y^{t+1} to be periodic, since $l_{ij} \geqslant 2m^2$. Having obtained a y^t which is periodic and satisfies $y_i^t = y_j^t = 0$, each subsequent y^t also enjoys these two properties (just ike for the x^t) and, in particular, y^m does. Therefore, if we compute y^m for all possible choices of l_i and l_j (at most m^2 of them), we can conclude that there is a periodic point y with $y_i = y_j = 0$ iff at least one of the m^2 periodic points computed satisfies this property. Thus a large Ω-circuit with a few *AND* and *OR* on top decides whether the clause $x_i \vee x_j$ can be included. Similar circuits give all the clauses for the 2-SAT description of the set of period points.

A similar onstruction gives the isomorphism g (via the permutation σ) of Theorem 4.1. All we need to find is two periodic points x and y that differ in coor-

dinate i, determine the coordinate j in which $f(x)$ and $f(y)$ differ, and conclude that $\sigma(j)=i$. Given a periodic point x^0, say with $x_i=0$, we can set $x^{t+1}=f^{(m_i)}(h_i(x^t))$ as before, where m_i is a multiple of $2l_i$ greater than m^2 and h_i changes the ith bit of its argument to a 1, until we obtain the point x^m. We can show as before that if a periodic point y with $y_i=1$ exists then $x_i^m=1$, and in fact x^m will be adjacent to a periodic point $x^m \oplus e_i$ with the ith coordinate equal to 0. Doing this for each i and for all possible cycle lengths l_i gives the isomorphism g. (We are not performing all these computations, only setting-up an Ω-circuit whose outputs, after a simple $\mathcal{NC}^1$ computation, will provide the description of g.)

This completes the $\mathcal{NC}^1$ reduction from problem (e) to (a), and from now on we can forget the function f and work with the succint description for the set of periodic points and for the behaviour of f provided by (e). If $g(x_1 \cdots x_m)= a \oplus x_{\sigma(1)} \cdots x_{\sigma(m)}$, then the fixed points of f must satisfy the additional constraints $x_j = a_j \oplus x_{\sigma(j)}$. We ensure this by collapsing x_j and $x_{\sigma(j)}$ into a single variable (after complementing one of them if $a_j=1$; the total number of such complementations in a cycle of σ must be even, otherwise we can conclude that there is no fixed point). This solves problem (d) from problem (2), providing a 2-SAT instance I' for the fixed points.

All that remains to be done is to obtain one fixed point in $\mathcal{NC}^1$. By Lemma 7.3, this can be done if the 2-SAT instance for the set of fixed points that we have constructed is transitively closed. The 2-SAT instance I that we had constructed for the periodic points was transitively closed, because it contained *all* the clauses that were satsfied by all periodic points. We must show that the 2-SAT instance remains transitively closed when variables in a cycle of a σ are merged into a single variable.

We mentioned before that in order for a fixed point to exist, the total number of complementatons in g for coordinates i in a cycle of σ has to be even (a complementation occurs when $y_i=1$ in the definition of g above). Conversely, it can be shown that when g is of this form, the mapping f must have a fixed point; in fact, as we saw in the proof of Theorem 5.1, every minimal fixed cube is then a fixed point. Assume w.l.o.g. that the fixed point of f is the all-zero bit vector $x=0$. Then g can be written in the form $g(x_1 \cdots x_m)=x_{\sigma(1)} \cdots x_{\sigma(m)}$.

We now claim that any partial solution to I' (i.e., an assignment to a subset S of the variables in I' that satisfies all the clauses involving only variables in S) can be extended to a complete solution. A partial solution to the instance I' for the fixed points can be viewed as a partial solution to the instance I for the periodic points, under the correspondence between the collapsed variables in I' and the cycle of variables in I that each collapsed variable corresponds to. This partial solution can be extended to a complete solution x to I because I is transitively closed (Lemma 7.1). Let x be a zero-most such periodic point. If x is not a fixed point, then the point $\text{med}(x, f(x), 0)$ is another extension of the partial solution to a complete solution which has a larger number of coordinates set to 0, a contradiction. Thus x is a fixed point. This shows that a partial solution to I' can be extended to a complete solution. This in turn implies that I' is transitively closed, because if there were a missing implied clause $u_i \vee u_j$ (where u_i and u_j are collapsed variables

corresponding to the variables in a cycle in σ), then $u_i = u_j = 0$ would be a partial solution that could be extended to a complete solution u, and the clause $u_i \vee u_j$ would in fact be an invalid clause. Therefore I' is transitively closed, and an $\mathcal{NC}^1$ circuit whose inputs are the clauses of I' provides a solution to I', i.e., a fixed point. ∎

The sizes of the circuits in the above constructions can be considerably reduced with some care, particularly if we only want a single fixed point, in which case we only need to obtain a 2-SAT instance for a subset of the periodic points, and even more if we only want to decide whether a fixed point exists, since this only depends on the parity of the number of complementations in the cycles of σ. To obtain a fixed point, we choose a path $x^0, ..., x^k$, where $x^k = f(x^0)$, then define $x^{i+k} = f(x^i)$. The periodic points $x^{m^3}, ..., x^{m^3+m^2}$ then form a path (and hence a connected set), and just by taking all the 2-SAT clauses satisfied by these points we obtain a connected median set S closed under f (here m^2 is just an upperbound for products $l_i l_j$ of cycle clengths of σ); these points also give us σ on the coordinates i that are non-constant in S (by connectedness of the path, some two points must differ only in i, and their images give us a j such that $\sigma(j) = i$). This gives a transitivley closed 2-SAT instance for fixed points in S which can then be solved in $\mathcal{NC}^1$, and Theorem 4.2 guarantees that if f has a fixed point then f will have a fixed point in S. Note that this algorithm is much simpler, since it does not require guessing all the possible l_i and l_j. It produces circuits of depth $O(m^3)$ and size $O(m^5)$. In the case of constant-size *scatter-free* gates, we can obtain $\mathcal{NC}^1$ reductions that produce $O(m)$ depth, $O(m^3)$ size Ω-circuits for constructing a stable configuration, matching the bound of Mayr and Subramanian [36] for the decision problem. All the constructions that we know require finding the 2-SAT description of a subset of the periodic points together with the behaviour of f in that subset, before we can focus on a particular solution. This is in sharp contrast with the situation in the case of monotone functions, where a zero-most fixed point exists and can be directly obtained.

For sets of monotone gates, the equivalence between (a) and (c) was known (in this case a stable configuration always exists so (b) is easy), and the preceding proof also shows equivalence with (d) and (e). The equivalence between (a) and (b) was known in the roommate case, where Ω consists of an X-gate. The equivalence between (b) and (c) in the roommate case, and the case of more general adjacency-preserving gates, were previously open. See [36] for more details.

10. Extensions and the Isometric Embedding Theorem

If the values assigned to the edges of the network are not restricted to just two possible values, then it is no longer clear how "limited fanout" should be defined. Here are some examples. The generalized X-gate takes two inputs from two finite sets U and V with two representative elements $u_0 \in V$; it produces two outputs from

the same two sets: if the inputs are u, v, then the outputs are u, v if either $u = u_0$ or $v = v_0$, and u_0, v_0 otherwise. On can check that changing the value of one of the inputs arbitrarily only changes the value of at most one output. The *average-of-three* gate takes three inputs a, b, c in the integer range $0, ..., k$, and produces three outputs $\lfloor (a+b+c)/3 \rfloor$, $\lfloor (a+b+c+1)/3 \rfloor$, and $\lfloor (a+b+c+2)/3 \rfloor$, in the same range. Changing the value of one of the inputs can now affect all three outputs. However, if we only change one of the input values by at most 1, then only one of the outputs changes, and it changes by at most 1, so in a sense fanout is limited. We can also define this gate for the case where the inputs are integers modulo $3k$, and the outputs are integers modulo k; then again changing an input by at most 1 modulo $3k$ chages an output by at most 1 modulo k, also suggesting that there is limited fanout.

Generalizing from the above examples, we associate a graph G_i with each edge in the network. These graphs will be cliques in the case of the generalized X-gate, paths of length k for the average-of-three gate, and cycles of length $3k$ or k respectively for the inputs and outputs of the modular average-of-three gate (we can also add self-loops). The possible values for an input or an output to a gate are then the vertices of the graph associated with the corresponding input or output edge. A gate is *adajacency-preserving* if, when two input words x and y differ only in the value of at most one coordinate, $x_i \neq y_i$, and the vertices x_i, y_i are adjacent vertices in the graph for input edge i, then the corresponding output words x' and y' also differ only in at most one coordinate, $x'_j \neq y'_j$, and the vertices x'_j, y'_j are adjacent vertices in the graph of output edge j.

A configuration of a network of adjacency-preserving gates is a vector $x = x_1 \cdots x_m$, where x_i is a vertex in the graph G_i corresponding to edge i. We view x as a vertex in the cartesian product $G = G_1 \square \cdots \square G_m$. This cartesian product is the graph with vertices $x = x_1 \cdots x_m$, $x_i \in G_i$, with an edge between $x = x_1 \cdots x_m$ and $y = y_1 \cdots y_m$ iff x_i is adjacent to y_i in G_i for some $1 \leqslant i \leqslant m$, and $x_j = y_j$ for $j \neq i$. If we define the transition function f of a network as before, then the condition that the gates be adjacency-preserving corresponds again of f being and edge-preserving mapping on the cartesian product G, i.e., if x and y are adjacent in G, then $f(x)$ and $f(y)$ are adjacent in G. The stable configurations of the network are the fixed points of f.

The algorithm of Section 3 does not seem to generalize to the case of an arbitrary edge-preserving mapping on an arbitrary cartesian product. The simplest example where we encounter difficulties is the cartesian product of 5-cycles (or 5-cycles with self-loops). The intuitive reason is that it is not clear in *pursuit* whether x should chase its image y in one direction or in the opposite direction in the cycle. We thus abandon this algorithm entirely and look for suitable generalizations for the theorems of Section 4. It turns our that much of the structure is preserved, and this leads to polynomial-time algorithms.

We shall define a generalization of median subgraphs of hypercubes to arbitrary graphs, namely the 2-*isometric* subgraphs. Let us first consider the notion of an *isometric* subgraph. A subgraph H of a graph G is an isometric subgraph if for

every pair of vertices x, y in H, the distance between x and y in G is equal to the distance between x and y in H. (In other words, we can walk along a shortest path in G from x to y without leaving the subgraph H.) If we denote the distance function in G by $\mathrm{dist}(x, y)$, then one way to state the fact that H is an isometric subgraph of G is to say that for all vertices x, y in H, if some vertex z is a neighbor of x in G and $\mathrm{dist}(z, y) = \mathrm{dist}(x, y) - 1$, then there exists such a z with z a neighbor of x in H as well. This motivates the definition of a *2-isometric* subgraph H: For all vertices x, y^1, y^2 in H, if some vertex z is a neighbor of x in G and $\mathrm{dist}(z, y^1) = \mathrm{dist}(x, y^1) - 1$, $\mathrm{dist}(z, y^2) = \mathrm{dist}(x, y^2) - 1$, then there exists such a z with z a neighbor of x in H as well. In other words, if we can walk from x towards y^1 and y^2 simultaneously in G, then we can also walk from x towards y^1 and y^2 simultaneously without leaving H.

Retracts are 2-isometric subgraphs. Recall that an edge-preserving mapping h on a graph G is called a *retraction* if $h^{(2)} = h$, in which case the graph $H = h(G)$ is called a *retract*. Now, given vertices x, y^1, y^2 is a retract H with a corresponding retraction h, if a neighbor of x in G and $\mathrm{dist}(z, y^1) = \mathrm{dist}(x, y^1) - 1$, $\mathrm{dist}(z, y^2) = \mathrm{dist}(x, y^2) - 1$, then $h(x) = x$, $h(y^1) = y^1$, $h(y^2) = y^2$, and $h(z)$ is a neigbor of x in H with $\mathrm{dist}(h(z), y^1) = \mathrm{dist}(x, y^1) - 1$, $\mathrm{dist}(h(z), y^2) = \mathrm{dist}(x, y^2) - 1$. Thus H is a 2-isometric subgraph of G. If H is the subgraph induced by the periodic points of an edge-preserving mapping f, then $f^{M!}$ is a retraction (where M is the number of vertices in G) and H is the corresponding retract. Thus periodic points form a 2-isometric subgraph.

The retract condition is in general a much stronger property than the 2-isometric subgraph condition. However, for our purposes, the notion of a 2-isometric subgraph is sufficiently strong. If H is a vertex-induced subgraph of a product graph $G = G_1 \square \cdots \square G_m$, then we define the *projection* $p_S(H)$ of H in the cartesian product G' of the G_i with $i \in S$ to be the vertex-induced subgraph of G' whose vertices can be obtained from vertices $x = x_1 \cdots x_m$ in H by discarding the coordinates x_i with i not in S. The resulting vertex x' is denoted by $p_S(x)$. If $S = \{i, j\}$ then we denote $p_S(H)$ by $p_{ij}(H)$ (we allow $i = j$). Note that $G' = p_S(G)$. The following is proved in [13].

LEMMA 10.1. *Let H be a vertex-induced subgraph of a product graph G. If H is a 2-isometric subgraph of G, then $p_S(H)$ is a 2-isometric subgraph of $p_S(G)$ for all S. Furthermore, H contains all the vertices x in G such that for all i, j, $p_{ij}(x)$ is in $p_{ij}(H)$, and $p_{ij}(H)$ is a 2-isometric subgraph of $p_{ij}(G)$ for all i, j, then H is a 2-isometric subgraph of G.*

This lemma gives a 2-SAT-like characterization of the 2-isometric subgraphs of product graphs; i.e., every such subgraph is characterized by conditions on pairs $\{i, j\}$ of coordinates. In particular, this implies that the set of periodic points in an adjacency-preserving network has a description of size polynomial in the size of the input (i.e., polynomial in the sume of the sizes of the graphs associated with each edge in the network). Note that in the case of the hypercube, the 2-isometric subgraphs are precisely the median subgraphs.

A vertex-induced subgraph H of a product graph $G = G_1 \square \cdots \square G_m$ is an *irredundant* subgraph if $p_i(H) = p_i(G)$ and $p_i(G)$ has at least two vertices, for all $1 \leqslant i \leqslant m$. Note that it is always possible to discard the vertices of $p_i(G)$ that are not in $p_i(H)$ and the dimensions i, where $p_i(G)$ has only one vertex, thus making H an irredundant subgraph of G. It is shown in [13] that if H is an irredundant 2-isometric subgraph of G, then H is a k-isometric subgraph of G for all $k \geqslant 1$, where a k-isometric subgraph is defined by extending the definition of 2-isometric subgraphs to the case of $k+1$ points $x, y^1, y^2, \ldots, y^k$ (thus 1-isometric is the same as isometric). This shows that the notion of a 2-isometric subgraph of a cartesian product is quite robust. Graham and Winkler proved an elegant result about the representation of graphs as isometric subgraphs of product graphs.

THEOREM 10.1 (Graham and Winkler [19, 48]). *Given a graph H, there exists a canonical embedding Φ of H as an irredundant isometric subgraph of a product graph $G = G_1 \square \cdots \square G_m$, in the following sense: Given any embedding Φ' of H as an irredundant isometric subgraph of a product graph $G' = G'_1 \square \cdots \square G'_{m'}$, the index set $\{1, \ldots, m\}$ can be partitioned into m' disjoint nonempty sets $S_1, \ldots, S_{m'}$ and each G'_i embedded as an irredundant isometric subgraph of the corresponding $p_{S_i}(G)$, so that the embedding of H into $G = p_{S_1}(G) \square \cdots \square p_{S_{m'}}(G)$ obtained from Φ' in this manner is precisely the canonical embedding Φ (after putting the m factors G_i if G back in the proper order).*

Note that the result is similar to the unique prime factorization theorem for cartesian products; in fact, the prime factorization theorem can be derived from it [48].

Suppose now that f is an isomorphism of a graph H and that H is canonically embedded as an irredundant isometric subgraph of a product graph $G = G_1 \square \cdots \square G_m$. Here Φ is simply the identity mapping. Then f can be viewed as another embedding Φ' if H as an irredundant isometric subgraph of G. Here $G' = G$, $G'_i = G_i$, and $m' = m$. By the theorem, we can partition $\{1, \ldots, m\}$ into m disjoint nonempty sets $S_i = \{\sigma(i)\}$ and define isomorphisms g_i^{-1} from $G'_i = G_i$ to $p_{S_i}(G) = G_{\sigma(i)}$, so that if $\Phi'(x) = y_1 y_2 \cdots y_{m'}$ with y_i in $G'_i = G_i$ and $x_{\sigma(i)} = g_i^{-1}(y_i)$ is in $p_{S_i}(G) = G_{\sigma(i)}$, then $\Phi(x) = x_1 x_2 \cdots x_m$. But $\Phi' = f$ and Φ is the identity mapping, so $f(x_1 \cdots x_m) = y_1 \cdots y_m = g_1(x_{\sigma(1)}) \cdots g_m(x_{\sigma(m)})$.

Given an edge-preserving mapping f on a product graph $P = P_1 \square \cdots \square P_{m'}$, if H is the subgraph of P induced by the periodic points of f, then H is an irredundant isometric subgraph of $H' = p_1(H) \square \cdots \square p_{m'}(H)$ after the factors $p_i(H)$ consisting of a single vertex have been discarded. Then H is a canonical irredundant isometric subgraph of a graph $G = G_1 \square \cdots \square G_m$ and H' can be viewed as a subgraph of $G = p_{S_1}(G) \square \cdots \square p_{S_{m'}}(G)$. We assume that the coordinates $1 \leqslant i \leqslant m$ of G have been labelled so that S_j contains smaller coordinates that $S_{j'}$ for $j < j'$. If $p_i(H)$ consists of a single vertex, we let $S_i = \{\}$ by convention. Since f restricted to H is an isomorphism of H, the argument of the last paragraph shows that there exists an isomorphism g of G, given by $g(x_1 \cdots x_m) = g_1(x_{\sigma(1)}) \cdots g_m(x_{\sigma(m)})$, such that $f(x) = g(x)$ for all x in H. We have thus shown:

COROLLARY 10.1. *Given an edge-preserving mapping f on a product graph $P = P_1 \cdots P_{m'}$, the mapping f restricted to the set of periodic points (the vertex set of a vertex-induced subgraph H) can be extended to an isomorphism $g(x_1 \cdots x_m) = g_1(x_{\sigma(1)}) \cdots g_m(x_{\sigma(m)})$ of a product graph $G = G_1 \square \cdots \square G_m$, where G is obtained from P by representing each $p_i(H)$ as a canonical irredundant isometric subgraph of a product $p_{S_i}(G)$.*

Thus an additional complexity has been introduced with respect to Theorem 4.1, because the isomorphism g is not defined on P but on a graph obtained from P by first discarding some vertices from each factor, then representing each factor as a subgraph of a product. In the case of the hypercube, discarding some vertices only makes H' be the smallest subcube of P containing all the periodic points, and then $G = H'$ because the factors of H' (single edges) are irreducible, so no additional, "imaginary" points are needed to obtain the graph G in the theorem; we thus obtain an isomorphism of the subcube H' which easily extends to an isomorphism of P, proving Theorem 4.1. In the general scenario of this section, we must be careful not to expect g to be defined in all of P or f to be defined in all of G.

Despite the additional complexity, we have obtained what we needed, i.e., a polynomial-size description for the behaviour of f on the set of periodic points given by the isomorphism g, which combined with the 2-SAT-like description for the set of periodic points gives us a good handle on periodic points and fixed points.

There is one additional requirement that we would like to satisfy in order to have a good characterization for periodic points. The graph H of periodic points is a 2-isometric subgraph of P, and therefore of H' as well, so it can be described by a 2-SAT-like instance on the product H'. Unfortunately, there may not be, in general, an isomorphism of the product H' that coincides with f in H. In fact, we need to extend H' to a product G by refining each of its factors to be able to obtain an isomorphism g on the extended graph G that coincides with f in H. However, the graph H may now no longer be a 2-isometric subgraph of G, so we have lost the 2-SAT-like characterization for H within the new product G. A natural question arises: is there an interpediate product graph G', with more factors than H' but fewer factors than G, such that we have H as a 2-isometric subgraph of G' and, at the same time, f restricted to H coincides with an isomorphism g' of G'? The answer is "yes." It is shown in [13] that the isometric embedding therem holds for 2-isometric subgraphs as well.

THEOREM 10.2. *Both Theorem* 10.1 *and Corollary* 10.1 *still hold when* isometric *is replaced by* 2-isometric.

This means that the attractor set H of the mapping f defined on the product P is a 2-isometric subgraph of P, of H', and of G as well; furthermore, there is an isomorphism g on G given by a permutation on the factors of G and isomorphisms between the corresponding factors, with the property that g coincides with f on H. These facts are enough to derive analogues of Theorem 4.4 and Theorem 9.1. We omit the proofs.

THEOREM 10.3. *Let f be an edge-preserving mapping on a product graph $P = P_1 \square \cdots \square P_m$, and let $n = \sum_{1 \leq i \leq m} |E(P_i)|$. Then for every vertex x in P, $f^{(k)}(x)$ can only be nonperiodic if $k < 10n^4$.*

THEOREM 10.4. *Let f be an edge-preserving mapping on a product graph $P = P_1 \square \cdots \square P_m$. Then one can find in time polynomial in the size of the input graphs P_i a succint description for the fixed point set of f and obtain from this description a particular fixed point. In fact, the fixed point problem can be reduced to a circuit value problem for an adjacency-preserving circuit over $\{f\}$, under $\mathcal{NC}^2$ reductions.*

11. FIXED PRODUCTS

Chung, Graham, and Saks [6] gave a complete characterization of the 2-isometric subgraphs of products of cliques (the *imprint-closed* subgraphs in their terminology) and showed that these subgraphs coincide with the retracts of products of cliques. We sketch a proof of these results.

By Lemma 10.1, a subgraph H of a product of cliques $G = K_1 \square \cdots \square K_m$ is a 2-isometric subgraph iff each of the projections $p_{ij}(H)$ is a 2-isometric subgraph of $p_{ij}(G)$, and the vertices of H are all the vertices x of G such that for all i, j, the vertex $x_i x_j$ is in $p_{ij}(H)$. We assume that H is an irredundant subgraph of G, i.e., $p_i(H) = p_i(G)$ has at least two vertices. It turns out that each $p_{ij}(H)$ has a very simple structure: either $p_{ij}(H) = p_{ij}(G)$, or $p_{ij}(H)$ contains precisely the vertices $x_i x_j$ in $p_{ij}(G)$ that satisfy $x_i = a \vee x_j = b$ for some values a, b. To see this, note that by connectivity, $p_{ij}(H)$ must contain two points ab_1 and ab_2 with $b_1 \neq b_2$. For every b' in $p_j(H) = p_j(G)$ there is some point $a'b'$ in $p_{ij}(H)$; the point ab' must then also be in $p_{ij}(H)$ (just let $x = a'b'$, $y_1 = ab_1$, $y_2 = ab_2$ in the definition of 2-isometric subgraphs). Therefore, there is a value a such that ab' is in $p_{ij}(H)$ for all b'. Similarly, there is a value b such that $a'b$ is in $p_{ij}(H)$ for all a'. So $p_{ij}(H)$ contains at least all $x_i x_j$ that satisfy $x_i = a \vee x_j = b$, for some a, b. It it contains some other $a'b'$, with $a' \neq a$ and $b' \neq b$, then we know from the preceding argument that containing $a'b$ and $a'b'$ for two different values b, b' implies containing $a'b''$ for all b''; but then $p_{ij}(H)$ contains for each b'' both ab'' and $\acute{a}b''$, for two different values a, a', so it must contain $a''b''$ for all a'', and $p_{ij}(H) = p_{ij}(G)$.

Therefore, for each i, j, either all values $x_i x_j$ are allowed, or $x_i x_j$ must satisfy a single constraint $x_i = a \vee x_j = b$. Note that if the constraint for i, j is $x_i = a \vee x_j = b$ and the constraint for j, k is $x_j = b' \vee x_k = x_k = c$, with $b \neq b'$ then i, k must satisfy $x_i = a \vee x_k = c$. We say that this third constraint *follows* from the previous two. An important property that can be easily checked is that the graph with an edge from each condition to the conditions that follow from it is acyclic. We can thus build a list L with all the constraints $x_i = a \vee x_j = b$, so that if constraint C_1 follows from constraint C_2, then C_1 *precedes* C_2 C_2 in this list.

There are several uses for this list L. One of them is the construction of a retrac-

tion h that has H as a retract. To each condition $x_i = a \vee x_j = b$ we can associate a generalized X-gate X_{ij}^{ab} which on input x produces output y with $y_k = x_k$ for $k \neq i, j$, and $y_i y_j = x_i x_j$ unless $x_i \neq a$ and $x_j \neq b$, in which case $y_i y_j = ab$. The retraction h is then the composition of the gates X_{ij}^{ab} corresponding to the constraints $x_i = a \vee x_j = b$, with each of these gates applied to the input x in turn, in the order that the corresponding constraints occur in L. (The idea is that each gate forces the corresponding constraint to hold.) This shows that all the 2-isometric subgraphs of products of cliques are actually retracts of that product.

We can also use this list L to define the *heart* Q of a 2-isometric subgraph H of a product of cliques G. The subgraph Q is defined by an iterative process. First, we find all the coordinates i and values a such that L contains a constraint of the form $x_i = a \vee x_j = b$ but no constraint of the form $x_i = a' \vee x_j = b$ with $a' \neq a$. We then set $x_i = a$ for all such (i, a) and eliminate all the constraints of the form $x_i = a \vee x_j = b$ from L. We now repeat the process of finding all coordinates i and values a such that x_i only occurs as $x_i = a$ in the updated list L, setting $x_i = a$ and eliminating the corresponding constraints $x_i = a \vee x_j = b$. The process must terminate with L empty. To see this, note that if L is not empty, and the first constraint $x_i = a \vee x_j = b$ in L is uch that coordinate j cannot be eliminated, then there must be some other constraint $x_j = b' \vee x_k = c$ in L with $b' \neq b$; but then there would be a constraint $x_i = a \vee x_k = c$ preceding the constraint $x_i = a \vee x_j = b$ in L, a contradiction. When L becomes empty, if we allow the x_i that were not fixed by setting $x_i = a$ for some a to take arbitrary values, we obtain a sub-product Q of G as a vertex-induced subgraph; furthermore, the sub-product Q is a subgraph of H. We call Q the *heart* of H.

THEOREM 11.1. *If H is an irredundant 2-isometric subgraph of a cartesian product of cliques $G = K_1 \square \cdots \square K_m$, then every isomorphism of H leaves the heart Q of H fixed.*

Proof. The embedding of H in G is the canonical embedding of Theorem 10.1, and this implies, by the discussion following that theorem, that the isomorphism of H can be extended to an isomorphism g of G. The mapping g is of the form $g(x_1 \cdots x_m) = h_1(x_{\sigma(1)}) \cdots h_m(x_{\sigma(m)})$. Since g is an isomorphism on H, the conditions $x_i = a \vee x_j = b$ in L must map to conditions $x_{\sigma(i)} = h_i^{-1}(a) \vee x_{\sigma(j)} = h_j^{-1}(b)$ also in L and vice versa, so the pair (i, a) is used in the above definition of the heart to eliminate x_i at the same time as the corresponding pair $(\sigma(i), h_i^{-1}(a))$ is used to eliminate $x_{\sigma(i)}$. This means that all the vertices in Q satisfy $x_i = a$ iff all the vertices in Q satisfy $x_{\sigma(i)} = h_i^{-1}(a)$; so g leaves Q fixed. ∎

The periodic points of an edge-preserving mapping on a graph define a retract, and retracts are 2-isometric subgraphs. On the set of periodic points, the edge-preserving mapping is an isomorphism. The preceding theorem then gives:

COROLLARY 11.1. *For every edge-preserving mapping f on a cartesian product of cliques G there is a sub-product Q (i.e., a cartesian product of subgraphs $p_i(Q)$ of the corresponding $p_i(G)$) such that $f(Q) = Q$.*

We have recently extended this result to products of arbitrary graphs. The proof uses the *distance center* of the attractor set as the fixed sub-product [14]. One disadvantage of that construction is that, unlike the heart, which is easily computed from the description of the attractor set, the distance center is $\#\mathcal{P}$-hard to find. On the other hand, we can still find a fixed sub-product in polynomial time, but then the fixed product depends on the edge-preserving mapping itself rather than on the structure of the attractor set alone. It seems that extending the notion of a heart to arbitrary product graphs would require not only using the 2-SAT-like characterization of the attractor set, but extending the inference structure of 2-SAT that allows us to infer certain clauses from other clauses, as well as the associated acyclic property that allowed us to define a (linear or partial) order L.

12. Conclusions and Open Problems

We have presented a new, graph-theoretic approach to stability problems on networks of gates with limited fanout. At the heart of this approach is a study of edge-preserving mappings on product graphs, their fixed points, and their periodic points. This study has applications to stable matching: it gives a new interpretation for the structure of partial orders and rotations that had been previously found, showing that these two concepts correspond to 2-isometric subgraphs of product graphs and to isomorphisms on these subgraphs. It also leads to simpler, more efficient, and more general algorithms for various problems in stable matching. The problems include the efficient representation of stable pairs, the enumeration of stable assignments, and the "optimal" stable matching problem in both the marriages and the roommates case. In parallel complexity, the problem of constructing a stable roommate assignment is shown to be no herder than the problem of deciding whether a stable assignment exists, implying that the problem is complete for the class $\mathcal{CC}$ [36].

Our algorithms are based on a pursuit strategy. We believe that the study of the products and retracts of graphs [13, 14] may find applications in other pursuit and location problems, because these problems often define classes of graphs that are closed under the product and retract operations.

Several questions remain open.

1. Is there a faster algorithm for finding stable configurations in an adjacency-preserving network? Our algorithm runs in $O(m^3)$ time, while the algorithm for scatter-free networks [36] takes $O(m)$ time. Note that the algorithm described in Section 3, unlike the algorithm of Section 6, does not take advantage of the bounded gatewidth of a network. Perhaps a more sophisticated algorithm can improve the running time by limiting the gatewidth. On the other hand, even with the proposed algorithm, we do not even know of instances for which the algorithm would even take $\Omega(m^2)$ time.

2. Can be bound in Theorem 4.4 on the number of iterations needed for a point to fall inside the attractor set be reduced to $O(m^2)$ as in the acatter-free case, or is there some example exhibiting super-quadratic growth? Can a similar improvement be achieved in the generalization of this result to arbitrary product graphs (Theorem 10.3)? Such improvements would reduce the size of the circuits constructed in Theorem 9.1 and Theorem 10.4.

3. An edge-preserving mapping on a product graph always has a fixed sub-product. Is it possible to find such a fixed product in polynomial time from the description of the attractor set? A positive answer to this question might require a better understanding of how clauses can be combined to infer other clauses in the non-boolean case.

4. We would like to know of other combinatorial problems, besides the stable matching problems, which correspond to network stability problems over adjacency-preserving gates, but which require gates that are more complex than the comparator and the X-gate. We would also like to find natural ways of allowing a larger amount of fanout in a network, in order to see how an increase in fanout affects the complexity of stability problems.

ACKNOWLEDGMENTS

I am indebted to Ashok Subramanian, who introduced me to this general framework and encouraged me throughout this work. I also had valuable discussions with Dan Gusfield, who introduced me to several of the questions on stable matching, and my advisor Donald Knuth. I also had useful conversations with Fan Chung, Edith Cohen, Danny Dolev, Joan Feigenbaum, Andrew Goldberg, Ron Graham, Nathan Linial, Ernst Mayr, Moni Naor, Christos Papadimitriou, Shaibal Roy, Michael Saks, and Peter Winkler.

REFERENCES

1. R. K. AHUJA, J. B. ORLIN, AND R. E. TARJAN, "Improved Time Bounds for the Maximum Flow Problem," Technical Report CS-TR-118-87, Department of Computer Science, Princeton University, 1987; *SIAM J. Comput.*, to appear.
2. H. J. BANDELT, Retracts of hypercubes, *J. Graph Thery* **8** (1984), 501–510.
3. H. J. BANDELT AND M. VEL, A fixed cube theorem for median graphs, Discr. Math. **67** (1987), 129–137.
4. R. BAR-YEHUDA AND S. EVEN, A linear time approximation algorithm for the weighted vertex cover problem, *J. Algorithms* **2** (1981), 198–203.
5. F. R. K. CHUNG, R. L. GRAHAM, AND M. E. SAKS, Dynamic search in graphs, *Discrete Algorithms Complexity* (1987), 351–387.
6. F. R. K. CHUNG, R. L. GRAHAM, AND M. E. SAKS, A dynamic location problem for graphs, *Combinatorica* **9** (1989), 111–132.
7. K. CLARKSON, A modification of the greedy algorithm for vertex cover, *Inform. Process. Lett.* **16** (1983), 23–25.
8. S. A. COOK AND M. LUBY, A simple parallel algorithm for finding a satisfying truth assignment to a 2-CNF formula, *Inform. Process. Let.* **27** (1988), 141–145.
9. R. P. DILWORTH, A decomposition theorem for partially ordered sets, *Ann. of Math.* **51** (1950), 161–166.

10. Deleted in proof.
11. T. FEDER, A new fixed point approach for stable networks and stable marriages, *in* "Proceedings, 21st ACM Symp. on Theory of Computing, 1989," pp. 513–522.
12. T. FEDER, 2-satisfiability and network flow, *Algorithmica*, to appear.
13. T. FEDER, Representations in product graphs, *J. Graph Theory*, to appear.
14. T. FEDER, "Stable Networks and Product Graphs," Doctoral dissertation, STAN-CS-91-1362, Stanford, 1991.
15. D. GALE AND L. S. SHAPLEY, College admissions and the stability of marriage, *Amer. Math. Monthly* **69** (1962), 9–15.
16. A. V. GOLDBERG AND R. E. TARJAN, A new approach to the maximum flow problem, *in* "Proceedings, 18th ACM Symp. on Theory of Computing, 1986, pp. 136–146.
17. A. V. GOLDBERG AND R. E. TARJAN, "Finding Minimum-Cost Circulations by Successive Approximation," Technical Report MIT/LCS/TM-333, Laboratory for Computer Science, MIT, 1987; *Math. Oper. Res.*, to appear.
18. R. L. GRAHAM, A mathematical study of magnetic domain interactions, *Bell Syst. Tech. J.* **49**, No. 8 (1970), 1627–1644.
19. R. L. GRAHAM AND P. M. WINKLER, On isometric embeddings of graphs, *Trans. Amer. Math. Soc.* **288**, No. 2 (1985), 527–536.
20. D. GUSFIELD, personal communication.
21. D. GUSFIELD, Three fast algorithms for four problems in stable marriage, *SIAM J. Comput.* **16**, No. 1 (1987), 111–128.
22. D. GUSFIELD, The structure of the stable roommate problem: Efficient representation and enumaration of all stable assignment, *SIAM J. Comput.* **17**, No. 4 (1988), 742–769.
23. D. GUSFIELD AND R. W. IRVING, The parametric stable marriage problem, *Inform. Process. Lett.*, to appear.
24. D. GUSFIELD AND R. W. IRVING, "The Stable Marriage Problem: Structure and Algorithms," MIT Press Series in the Foundations of Computing, MIT Press, Cambridge, MA, 1989.
25. D. GUSFIELD, R. IRVING, P. LEATHER, AND M. SAKS, Every finite distributive lattice is a set of stable matchings for a small stable marriage, *J. Combin. Theory Ser. A* **44** (1987), 304–309.
26. D. GUSFIELD AND L. PITT, Equivalent approximation algorithms for node cover, *Inform. Process. Lett.* **22**, No. 6 (1986), 291–294.
27. D. GUSFIELD AND L. PITT, "A Bounded Approximation for the Minimum Cost 2-SAT Problem," Technical Report CSE-89-4, University of California, Davis, 1989.
28. P. HELL, Retracts in graphs, *in* "Lecture Notes in Math., Vol. 406," pp. 291–301, Springer-Verlag, Berlin, 1974.
29. P. HELL AND I. RIVAL, Absolute retracts and varieties of reflexive graphs, *Cand. J. Math.* **39**, No. 3 (1987), 544–567.
30. D. S. HOCHBAUM, Approximation algorithms for the set covering and vertex cover problems, *SIAM J. Comput.* **11**, No. 3 (1982), 555–556.
31. R. W. IRVING, An efficient algorithm for the stable roommates problem, *J. Algorithms* **6** (1985), 477–595.
32. R. W. IRVING, P. LEATHER, AND D. GUSFIELD, An efficient algorithm for the optimal stable marriage, *J. Assoc. Comput. Mach.* **34**, No. 3 (1987), 532–543.
33. R. W. IRVING AND P. LEATHER, The complexity of counting stable marriages, *SIAM J. Comput.* **15**, No. 3 (1986), 655–667.
34. R. KARP, E. UPFAL, AND A. WIGDERSON, Are search and decision problems computationally equivalent?, *in* "Proceedings, 17th ACM Symp. on Theory of Computing, 1985," pp. 22–32.
35. D. E. KNUTH, "Mariages stables et leur relations avec d'autres problèmes combinatoires," Les Presses de l'Université de Montréal, Montreal, 1976.
36. E. MAYR AND A. SUBRAMANIAN, The complexity of circuit value and network stability, *in* "Fourth Annual Conference on Structure in Complexity Theory, 1989," pp. 114–123; full version in Technical Report STAN-CS-89-1278, Stanford University, 1989.

37. M. MULDER, "n-cubes and median graphs," *J. Graph Theory* **4** (1980), 107–110.
38. G. L. NEMHAUSER AND R. E. TROTTER, Vertex packing structural properties and algorithms, *Math. Programming* **8** (1975), 232–248.
39. C. NG, Lower bounds for the stable marriage problem and its variants, *in* "Proceedings, 30th IEEE Symp. on Foundations of Computer Science, 1989," pp. 129–133.
40. R. NOWAKOWSKI AND I. RIVAL, Retract rigid cartesian products of graphs, *Discrete Math.* **70** (1988), 169–184.
41. G. PÓLYA, R. E. TARJAN, AND D. R. WOODS, "Notes on Introductory Combinatorics," Birkhäuser, Basel, 1983.
42. J. S. PROVAN AND M. O. BALL, The complexity of counting cuts and of computing the probability that a graph is connected, *SIAM J. Comput.* **12**, No. 4 (1983), 777–788.
43. A. QUILLIOT, "Homomorphismes, point fixes, rétractions et jeux de poursuite dans les graphes, les ensembles ordonnés et les espaces métriques," Thèse d'Etat, Université de Paris VI, Paris 1983.
44. E. RONN, "On the Complexity of Stable Matchings with and without Ties," Ph. D. thesis, Yale University, 1986.
43. A. SUBRAMANIAN, personal communication.
45. A. SUBRAMANIAN, "A New Approach to Stable Matching Problems," Technical Report STAN-CS-89-1275, Stanford University, 1989.
46. A. SUBRAMANIAN, "The complexity of circuit value and network stability," Doctoral dissertation, Stanford University, 1989.
47. R. E. TARJAN, Depth-first search and linear graph algorithms, *SIAM J. Comput.* **1** (1972), 146–160.
48. P. M. WINKLER, The metric structure of graphs: Theory and applications, *in* "Eleventh British Combinatorial Conference, 1987."

Printed in Belgium
Uitgever: Academic Press, Inc.
Verantwoordelijke uitgever voor België:
Hubert Van Maele
Altenastraat 20, B-8310 Sint-Kruis