International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:3, No:11, 2009

A Self-stabilizing Algorithm for Maximum Popular Matching of Strictly Ordered Preference Lists

Zhengnan Shi
Dept of Mathematics and Computer Science
University of Wisconsin Whitewater
Whitewater, WI, 53190
USA

Abstract—In this paper, we consider the problem of Popular Matching of strictly ordered preference lists. A Popular Matching is not guaranteed to exist in any network. We propose an ID-based, constant space, self-stabilizing algorithm that converges to a Maximum Popular Matching—an optimum solution, if one exist. We show that the algorithm stabilizes in $O(n^5)$ moves under any scheduler (daemon).

Keywords—self-stabilization, popular matching, algorithm, distributed computing, fault tolerance

I. Introduction and Related Work

Self-stabilization is a strong and desirable fault-tolerance property. The self-stabilizing approach is introduced by Dijkstra [3]. We define a distributed network as a connected, undirected graph G with node set V and edge set $E \subseteq V \times V$. Let $n = |V|$ and $m = |E|$. Two nodes joined by an edge are said to be *neighbors*. We use $N(i)$ to denote the set of neighbors of node i—its (open) *neighborhood*. The contents of a node's local variables are defined as its *local state*. The system's *global state* is the union of all local states. If you take an arbitrary distributed algorithm and start it in a state where its variables have been set to a random value from its domain, the behavior is usually not predictable. However, starting from *any* initial configuration and in *every* execution, self-stabilizing systems are required to recover to a set of legal states.

A. Self-stabilizing Algorithms

A self-stabilizing algorithm is presented as a set of rules, each with a boolean predicate and an action. The rules of our algorithms are of the form $p \rightarrow M$, where p is a Boolean *predicate*, and M is a *move* which changes local variable(s). A node is said to be *privileged* if the predicate p is true. If a node becomes privileged, it may execute the corresponding move M. In the shared-variable version of this paradigm, every node executes the same set of self-stabilizing rules, and maintains and changes its own set of local variables based on the current values of its variables and those of its *neighbors*. We assume that there exists a *daemon*, an adversarial oracle as introduced in [3], [11], which at each time-step selects one or more of the privileged nodes to move. In the *serial*, also known as the *central daemon* model, no two nodes move at the same time. In the *distributed daemon* model, the daemon can choose any subset of privileged nodes to move simultaneously. Self-stabilizing algorithms can be designed for networks that are either *ID-based* or for the networks that are *anonymous*. In an ID-based network, each node has a unique ID. In an anonymous network, the nodes lack unique IDs, so there is not a priori way of distinguishing them. It is known that, given IDs, any algorithm for the central daemon can be transformed into one for the distributed daemon (see for example [2]). For a complete discussion of self-stabilization, see the books by Dolev [4] or Tel [15].

email shiz@uww.edu, telephone (262)472-5006, fax (262)472-1372.

When no further state change is possible, we say that the system is in a *stable configuration*. A self-stabilizing algorithm must satisfy:

1) From any initial illegitimate state it reaches a legitimate state after a finite number of moves; and
2) For any legitimate state and for any move allowed by that state, the next state is a legitimate state.

The complexity of a self-stabilizing algorithm is measured by the upper bound of the number of moves and/or *rounds* [4], [5]. A *round* is the minimum period of time where every node that is continually privileged moves at least once.

Several graph problems arise naturally in distributed systems. For example, distributed algorithms for finding matchings, independent sets, dominating sets and colorings have been studied [10]–[14]. Given the limited power of the self-stabilizing paradigm, one usually expects to achieve optimality, and thus must settle for a good coloring, a maximal matching or a minimal dominating set. However, the proposed self-stabilizing algorithm is guaranteed to produce a Popular Matching of Maximum cardinality, an optimum solution, if one exists in the network. We show that the algorithm stabilizes in $O(n^5)$ moves under any scheduler (daemon).

B. the Popular Matching Problem

The concept of a *Popular Matching*, also known as a *Majority Assignment*, was first introduced by Gardenfors [7] in the context of the stable marriage problem [6], [8]. *Popular Matching problem* models some important real-world problems.

An instance of the Popular Matching problem is defined on a bipartite graph $G = (V = A \cup P, E)$ and a partition $E = \{E_1 \dot{\cup} E_2 \cdots \dot{\cup} E_r\}$ of the edge set. All the edges in E_i are called of rank i. The nodes in A are called *applicants*,

the nodes in P *posts*. The sets of applicants and posts are distinct, $A \cap P = \emptyset$. If $(a,p_1) \in E_i$ and $(a,p_2) \in E_j$ with $i < j$, we say that a prefers p_1 to p_2. If $i = j$, we say that a is *indifferent* between p_1 and p_2. This ordering of posts adjacent to a is called a's *Preference List*. Our paper considers the cases of *strictly-ordered* preference lists where no applicant is indifferent between any two posts on its preference list.

A *Matching* M of G is a set of edges no two of which share an endpoint. A node $i \in A \cup P$ is either *unmatched* in M, or *matched* to some node, denoted by $M(i)$ which means $(i, M(i)) \in M$. The two end nodes are one applicant and one post. Let $a \in A$ be an applicant, M_1 and M_2 be two matchings. There are two cases when we say applicant a *prefers* matching M_1 to matching M_2. Case one is when a is matched in M_1 and unmatched in M_2. Case two is when a is matched in both M_1 and M_2, and a prefers $M_1(a)$ to $M_2(a)$. We use symbol $\succ$ to denote the *more popular than* relationship between matchings. $M_1 \succ M_2$ if the number of applicants that prefer M_1 to M_2 exceeds the number of applicants that prefer M_2 to M_1. The *more popular than* relation is not acyclic. Therefore, a Popular Matching does not always exist in any networks. The definition of *Popular Mathcing* is given in [1].

Definition. *A matching M is a* Popular Matching *if and only if there is no matching that is* more popular than *M.*

II. A Prelude Algorithm

We first introduce a self-stabilizing algorithm which finds a Maximum Matching in Even Cycles ($MMEC$). It is part of the solution to Popular Matching. Algorithms $MMEC$ and Maximum Popular Matching (MPM) introduced in section 3 use the same mechanism in their design. Since $MMEC$ is significantly simpler, it leads up to MPM. We assume each node in the network has a unique ID. Without loss of generality, we let the ID's be between 1 and n (the order of the graph). When we say node i, variable i holds the ID of the node. Local variables and notations of Algorithm $MMEC$ are listed next.

1) Variable $c0$ is an unsigned integer, $0 \leq c0 \leq n$.
2) Variables $nx0$ (next) and $r0$ (root) point to neighbors. They hold either the ID's of nodes or 0. Subscription is used to denote the ID of the hosting node of a variable. Since an ID is at least 1, let $c0_0 = nx0_0 = r0_0 = null$. By design, variables $r0$ and $c0$ jointly identify every node matched along a path. A *matched path* is a monotonic and continuous integer sequence of the counter $c0$'s. It starts from a node (call it j) of $c0_j = 1$. Every node in the path has the same $r0$ which holds the ID of node j. The counter $c0$'s of any two adjacent nodes differ by 1. We call the end of the matched path with $c0 = 1$ the *lower end*. The other end is the *higher end*. Except for the node at the higher end, the variable $nx0_i$ of a node i holds the ID of the next node in the matched path.
3) Algorithm $MMEC$ runs in even cycles. Therefore, each node has 2 neighbors, call them x and y. For proposition, let $x > y$. Notation x/y means both x and y. For example, $r0_{x/y}$ means both $r0_x$ and $r0_y$.
4) Variable $mpe0(i)$ (matched path extension for $c = 0$) denotes a unique neighbor $j \in \{x,y\}$, such that $c0_j > 0 \wedge nx0_j = i \wedge r0_j \geq r0_{x/y} \wedge r0_j > r0_i \wedge r0_j > i$. If no neighbor satisfies this condition, $mpe0(i)$ is $null$. If both x and y satisfy this condition: (1) if there is a neighbor $j \in \{x,y\}$ such that $r0_j = j \wedge c0_j = 1$, then this neighbor j is $mpe0(i)$; (2) otherwise, $mpe0(i)$ is the neighbor with a larger $c0$; (3) if $c0_x = c0_y$, then $mpe0(i)$ is x.
5) For a node i with $c0_i > 1$, $cnbr0^-(i)$ denotes the set of neighbors $\{j | j \in \{x,y\} \wedge c0_j = c0_i - 1 \wedge r0_j = r0_i \wedge nx0_j = i \wedge nx0_i \neq j\}$.

The rules of our self-stabilizing algorithm are represented by a list of if-then statements. To make a rule concise, we omit the negation of the desired state (the then statement) from the condition. If the desired state is already achieved, the node shall not be privileged by the rule. Assume the algorithm runs on node i.

Algorithm 1: Maximum Matching in Even Cycles

Start Rule
  if $i > x/y \wedge i > r0_{x/y}$
    then $c0_i = 1 \wedge r0_i = i \wedge nx0_i = x$
Extension Rule
  if $mpe0(i) \neq null$
    then $c0_i = c0_{mpe0(i)} + 1 \wedge r0_i = r0_{mpe0(i)} \wedge nx0_i = k, k \in \{x,y\} \wedge k \neq mpe0(i)$
Clean Rules
(1) **if** $(c0_i = 1 \wedge (r0_i \neq i \vee nx0_i \neq x)) \vee c0_i = 0$
      then $c0_i = r0_i = nx0_i = 0$
(2) **if** $c0_i > 1 \wedge cnbr0^-(i) = \emptyset$
      then $c0_i = r0_i = nx0_i = 0$

Rules of Algorithm Maximum Matching in Even Cycles

Upon stabilization, the matches are identified by looking at each node i of an even counter $c0_i \geq 2$. If $cnbr0^-(i) \neq \emptyset$, then there is a match between node i and any node in $cnbr0^-(i)$. The correctness of Algorithm $MMEC$ is shown in the following lemma. The complexity of Algorithm $MMEC$ when executed with Algorithm MPM is proved in section 5.1.

Lemma 1: Every node is matched upon stabilization of Algorithm $MMEC$.

Proof:

By the Clean Rule 2, every node with $c0 > 1$ has $cnbr0^- \neq \emptyset$. If we allow a single node with $c0 = 1$ as a matched path, the cycle is divided into one or more matched paths. Assume there are more than 1 matched paths. Let i with $c0_i = 1$ and j with $c0_j = 1$ be the lower ends of two paths. Since each node has a unique ID, we assume $i > j$ without loss of generality. Let k be the node at the higher end of the matched path of i. Let l be k's neighbor that is not in the matched path of i. Neither k nor l is privileged by the Extension Rule. We have $r0_l > r0_k = i$. By symmetry, since $r0_k < r0_l$, node l must be the lower end of its matched path. Since the cycle is connected, the higher end of a matched path with $r0 > i$ will be adjacent to the matched path of j. The adjacent node in the matched path of j is privileged to move by the Extension Rule, a contradiction.

Hence, there is one matched path in the cycle. Every node is matched in the even cycle. ■

III. A Self-stabilizing Algorithm for Maximum Popular Matching

If a post p is the first-ranked post on an applicant's preference list, it is called an *f-post*. For an applicant a, $f(a)$ refers to the first-ranked post on its preference list, $(a, f(a)) \in E_1$. $s(a)$ denotes the first non-f-post on a's preference list. We call any such post p an *s-post*. An applicant has an edge to every post on its preference list in graph G. We define the *Reduced Graph* $G' = (A \cup P, E')$ as a subgraph of G containing at most two edges for applicant a: $(a, f(a))$ and $(a, s(a))$. If every post in a preference list is an f-post, then the applicant has degree one in the reduced graph G'. We assume a preference list has at least one post. Hence, there is no isolated applicant. For conciseness, we assume the reduced graph G' is connected. (Otherwise, our algorithm runs on each connected component.) For the rest of the paper, the network refers to the reduced graph G'.

A. Network Configuration

Every node has an invariant identifying whether it is an applicant or a post. Each post keeps an invariant f. If a node $p \in P$ is an f-post, then $f_p = true$. Otherwise, $f_p = false$. The subscription identifies the residing node of the variable. Algorithm Maximum Popular Matching (MPM) utilizes three local variables.

1) Variable c is an unsigned integer, $0 \leq c \leq n$.
2) Variables nx (next) and r (root) point to neighbors. They hold either the ID's of nodes or 0. Since an ID is at least 1, let $c_0 = nx_0 = r_0 = null$. We continue to use the concept of a *matched path* which is a monotonic and continuous integer sequence of the counter c's. It starts from a node (call it j) of $c_j = 1$. Every node in the path has the same r which holds the ID of node j.

To abridge the presentation of our algorithm, we use notations based on variables c, r and nx.

1) Algorithm MPM is concerned about the reduced graph G'. Let $N(i)$ denote the set of neighbors of i in G'. If $|N(i)| = 1$, $u(i)$ refers to the unique neighbor.
2) In the Clean Rule 4, $pufp$ denotes a set of *pointed unmatched f-posts* $\{i | i \in P \wedge f_i \wedge c_i > 2 \wedge nx_i \neq 0 \wedge (\forall j \in N(i), nx_j = i \wedge c_j \ is \ even \wedge (2 \leq c_j < n-1) \wedge r_j > 0)\}$.
3) Let $ps(i)$ (the proposer set) denote the set $\{j | j \in N(i) \wedge nx_j = i \wedge r_j > 0 \wedge c_j \ is \ odd \wedge c_j < n\}$.
   a) For an applicant a, if $f(a) \in ps(a)$, then $pp(a)$ (proposer) is $f(a)$. If $s(a)$ is the only node in $ps(a)$, then $pp(a)$ is $s(a)$. Let $pp(a) = null$ if $ps(a)$ is empty.
   b) For a post p, if $|ps(p)| = 1$, then $pp(p)$ is the one node in $ps(p)$. Let $pp(p) = null$ if $|ps(p)| \neq 1$.
4) For a node i with $c_i > 1$, $cnbr^-(i)$ denotes the set of neighbors $\{j | j \in N(i) \wedge c_j = c_i - 1 \wedge r_j = r_i \wedge nx_j = i \wedge nx_i \neq j\}$. $cnbr^1(i)$ represents the set $\{j | j \in N(i) \wedge c_j = 1 \wedge nx_j = i \wedge nx_i \neq j\}$. $cnbr^0(i)$ denotes the set $\{j | j \in N(i) \wedge c_j = 0 \wedge nx_i \neq j\}$. For clarification, $cnbr^0(i)$ is different from $N_0(i) = \{j | j \in N(i) \wedge c_j = 0\}$.
5) inm (initiating next match) denotes the set $\{i | i \in V \wedge c_i = 0 \wedge (\forall j \in N(i), c_j \ is \ even \wedge c_j \neq 0 \rightarrow nx_j = i)\}$.
6) Variable $mpe(i)$ (matched path extension) denotes a neighbor $j \in N(i)$, such that $nx_j = i \wedge c_j \ is \ even \wedge (2 \leq c_j < n-1) \wedge r_j > 0 \wedge (\forall k \in N(i) - \{j\}, c_k \neq 0 \rightarrow r_j > r_k)$. If there is no such node, $mpe(i)$ is $null$.
7) In the f Rule, ufp (the unmatched f-posts) denotes a set of *unmatched f-posts* $\{i | i \in P \wedge f_i \wedge c_i = 0 \wedge (\forall j \in N(i), nx_j = i \wedge c_j \ is \ even \wedge (2 \leq c_j < n-1) \wedge r_j > 0)\}$. If node $p \in ufp$, all the neighboring applicants are matched but not with p.
8) Our max function returns the largest ID (node) in $N(i)$. A condition q may be inserted in a pair of curly brackets following the max. For example, $\max\{j \neq 3\}$ returns the largest ID that is not 3 in $N(i)$.

Algorithm MPM consists of four groups of rules: the Start Rule, the Clean Rules, the Extension Rules and the f-Rule. The predicates of the rules are evaluated in the listed order. We assume the rules are executed at a node of ID i.

Algorithm 2: Maximum Popular Matching

Start Rule
  if $|N(i)| = 1$
    then $c_i = 1 \wedge r_i = i \wedge nx_i = u(i)$
Clean Rules
(1) **if** $c_i = 1 \wedge |N(i)| > 1$
    then $c_i = r_i = nx_i = 0$
(2) **if** $c_i > 1 \wedge cnbr^-(i) = \emptyset$
    then $c_i = r_i = nx_i = 0$
(3) **if** $c_i > 1 \wedge c_i \ is \ odd \wedge |cnbr^0(i)| + |cnbr^1(i)| > 0$
    then $c_i = r_i = nx_i = 0$
(4) **if** $c_i > 1 \wedge c_i \ is \ odd \wedge (r_{nx_i} \neq r_i \vee c_{nx_i} \neq c_i + 1) \wedge c_{nx_i} \neq 0 \wedge i \notin pufp$
    then $c_i = r_i = nx_i = 0$
Extension Rules
(1) **if** $pp(i) \neq null \wedge (c_i = 0 \vee (c_i > 1 \wedge i \in A \wedge j = f(i) \wedge r_j \neq r_i))$
    then $c_i = c_{pp(i)} + 1 \wedge r_i = r_{pp(i)} \wedge nx_i = 0$
(2) **if** $c_i \ is \ even \wedge c_i \geq 2 \wedge r_i > 0 \wedge nx_i = 0 \wedge \exists_1 l \in N(i), c_l = 0$
    then $nx_i = l$
(3) **if** $i \in inm \wedge mpe(i) \neq null \wedge \exists_1 l \in N(i), c_l = 0$
    then $c_i = c_{mpe(i)} + 1 \wedge r_i = r_{mpe(i)} \wedge nx_i = l$
f Rule
  if $i \in ufp \wedge mpe(i) \neq null$
    then $c_i = c_{mpe(i)} + 1 \wedge r_i = r_{mpe(i)} \wedge nx_i = \max\{j \neq mpe(i)\}$

Rules of Algorithm Maximum Popular Matching

Upon stabilization, the matches are identified by looking at each node i of an even counter $c_i \geq 2$. If $cnbr^-(i) \neq \emptyset$, then there is a match between node i and any node in $cnbr^-(i)$.

IV. Correctness

We prove that the algorithms MPM and $MMEC$ stabilize at a Maximum Popular Matching if one exists. The following lemma is cited from [1].

Lemma 2: M is a popular matching of network G if and only if (i) every f-post p is matched to an applicant in $f(p)$, and (ii) for each applicant a, $M(a) \in \{f(a), s(a)\}$.

Condition (ii) of Lemma 2 dictates that every Popular Matching is applicant-complete in the reduced graph G'. The case is trivial for a connected G' with less than 4 vertices. There is no matching if G' has one node. There is one matching in a connected G' of a post and an applicant; hence it is a Popular Matching. In the cases of three nodes, there is one

case which admits a Popular Matching. There is a (Maximum) Popular Matching of cardinality 1 if the reduced graph G' has one applicant a and two posts $f(a)$ and $s(a)$. Without special note, we assume $n \geq 4$.

Lemma 3: Let i and j be two nodes in the network. Upon stabilization, $r_i = r_j$ and $c_i = c_j$ cannot both be true.

Proof:

Assume there are two nodes i and j with $r_i = r_j$ and $c_i = c_j$. Since i and j are not privileged by the Clean Rule 2, $cnbr^-(i)$ and $cnbr^-(j)$ are not empty. Since a variable nx holds one ID, there are at least two nodes with $r = r_i$ and $c = c_i - 1$. Since the Clean Rule 2 requires $c > 1$, there are two nodes of $c = 1$ of the same ID. This contradicts the fact that each node has a unique ID. ■

Lemma 4: Upon stabilization, if a node i has $c_i > 1$, then $r_i > 0$ and there is a matched path from node i to a node of $c = 1$. If c_i is odd, then $c_i < n$. If c_i is even, then $c_i \leq n$.

Proof:

Let i be a node with $c_i > 1$. Because node i is not privileged by the Clean Rule 2, there must exist a node $j \in cnbr^-(i)$. Similarly, if $c_j > 1$, there must exist a node $k \in cnbr^-(j)$. Since the Clean Rule 2 requires $c > 1$, there is a matched path from node i to a node of $c = 1$, call it l. By the Start Rule, $r_l = l > 0$. Every node in the matched path has $r = r_l > 0$. By Lemma 3 the counter c of each node is unique, there are c_i nodes in the matched path. Hence $c_i \leq n$.

We use proof by contradiction to show $c_i < n$ if c_i is odd. Let $c_i = n$ and c_i is odd. If $nx_i = 0$, then $i \notin pufp$ by the definition of $pufp$. Since $c_0 = null$, node i is privileged by the Clean Rule 4. Hence, $nx_i \neq 0$. If $r_{nx_i} = r_i$ and $c_{nx_i} = c_i + 1$, then $c_i < n$ because $c_{nx_i} \leq n$, a contradiction. Let $r_{nx_i} \neq r_i$ or $c_{nx_i} \neq c_i + 1$. If $r_{nx_i} \neq r_i$ or $c_{nx_i} = 0$, then node nx_i is not in the matched path of i. Since the total number of nodes is n, $c_i < n$, a contradiction. The last case is $r_{nx_i} = r_i$, $c_{nx_i} \neq 0$ and $c_{nx_i} \neq c_i + 1$. If $i \notin pufp$, then node i is privileged by the Clean Rule 4. If $i \in pufp$, then i is a post. Because $c_i \neq 1$, node i has at least 2 neighbors. Since $c_i = n$, all neighbors of node i are in the matched path of i. In the reduced graph G', there is at most one edge between any two nodes. By Lemma 3, each counter has a unique value in the matched path. At least one neighbor, call it k, has $c_k \neq c_i - 1$. Node k is adjacent to $nx_k \neq i$ and i, hence $c_k > 1$. Also because k is not privileged by the Clean Rule 2, $cnbr^-(k) \neq \emptyset$. Hence, applicant k has degree at least 3, a contradiction. ■

Lemma 5: Upon stabilization, every node with an even counter $c > 1$ is matched.

Proof:

Let i be a node with an even counter $c_i > 1$. Node i is matched with a node in $cnbr^-(i)$. Since i is not privileged by the Clean Rule 2, $cnbr^-(i)$ is not empty. ■

Lemma 6: Upon stabilization, if a post p has $|ps(p)| > 1$, then the network does not have a Popular Matching.

Proof:

Since $|ps(p)| > 1$, at least one neighboring applicant, call it $a \in ps(p)$, cannot be matched to p. By the definition of $ps(p)$, c_a is odd. If $c_a = 1$, then node a is an unmatched applicant. Let $c_a > 1$, and the other neighbor of a be post q, c_q is even. By Lemma 4, there is a matched path from post q to an applicant of $c = 1$. In the matched path, every node is matched. Each matched applicant has at most one alternative post to match. If we force a match of a and q, another applicant along the matched path becomes unmatched. An applicant-complete matching does not exist in the network. By Lemma 2, there is no Popular Matching in the network. ■

Lemma 7: Upon stabilization, if a node with an odd counter c is adjacent to a neighbor of $c = 0$, then the network does not have a Popular Matching.

Proof:

Proof by contradiction. Let the counter c_i of a node i be odd. Let $j \in N(i)$ be a node with $c_j = 0$. If $c_i = 1$, then $r_i = i > 0$ and $nx_i = j$ by the Start Rule. Therefore $i \in ps(j)$. If $c_i > 1$, $|cnbr^0(i)| = 0$ because node i is not privileged by the Clean Rule 3. Hence $nx_i = j$. By Lemma 4, $r_i > 0$ and $c_i < n$. Hence $i \in ps(j)$. Since $i \in ps(j)$ and j is not privileged by the Extention Rule 1, $pp(j) = null$ and j must be a post with $|ps(j)| > 1$. By Lemma 6, there is no Popular Matching in the network, a contradiction. ■

Lemma 8: If a network admits a Popular Matching, every node with an odd counter $c > 1$ is matched upon stabilization.

Proof:

Let i be a node with an odd counter $c_i > 1$. By Lemma 4, $c_i < n$ and $r_i > 0$. If $nx_i = 0$, then $i \notin pufp$ by the definition of $pufp$. Since $c_0 = null$, node i is privileged by the Clean Rule 4. Hence, $nx_i \neq 0$.

If $i \in pufp$, then node i is $f(nx_i)$. If $r_{nx_i} \neq r_i$, then node nx_i is privileged by the Extension Rule 1. If $r_{nx_i} = r_i$ and $c_{nx_i} \neq c_i + 1$, node nx_i is in the same matched path. Note that the matched path cannot continue after node i, $c_{nx_i} < c_i$. By definition, $nx_i \notin cnbr^-(i)$. By Lemma 3, each counter has a unique value in the matched path, hence $c_{nx_i} \neq c_i - 1$. Because nx_i is adjacent to $nx_{nx_i} \neq i$ and i, $c_{nx_i} > 1$. Since nx_i is not privileged by the Clean Rule 2, $cnbr^-(nx_i) \neq \emptyset$. Hence, applicant nx_i has degree at least 3, a contradiction. Hence $r_{nx_i} = r_i$ and $c_{nx_i} = c_i + 1$. By Lemma 3, each counter has a unique value in the matched path. Hence, $|cnbr^-(nx_i)| = 1$, i is matched with nx_i. If $i \notin pufp$ and $c_{nx_i} \neq 0$, node i is matched, otherwise it is privileged by the Clean Rule 4. Finally, if $i \notin pufp$ and $c_{nx_i} = 0$, by Lemma 7 there is no Popular Matching in the network, a contradiction. ■

Lemma 9: If a network admits a Popular Matching, every node with $c > 0$ is matched upon stabilization.

Proof:

By Lemmas 5 and 8, every node with $c > 1$ is matched upon stabilization. Assume there is a node i with $c_i = 1$. By the Start Rule, $nx_i = u(i)$ and $r_i = i$. If $r_{u(i)} = i$, then $u(i)$ is in the matched path starting at i. Since $u(i)$ is the only node adjacent to i, $c_{u(i)} = c_i + 1 = 2$. Node i is matched with $u(i)$.

Let $r_{u(i)} \neq i$. Since $u(i)$ is not privileged by the Extention Rule 1 and $i \in ps(u(i))$, $u(i)$ must be a post with $|ps(u(i))| > 1$. By Lemma 6, there is no Popular Matching in the network, a contradiction. ■

Lemma 10: Upon stabilization, if a node i with $c_i > 1$ and c_i is even has $|cnbr^0(i)| > 0$, then i has degree at least 3, a post.

Proof:

Let j be a neighbor in $cnbr^0(i)$. $nx_i \neq j$ by the definition of $cnbr^0(i)$. By Lemma 4, $r_i > 0$. If $nx_i = 0$, then node i is adjacent to at least 2 nodes of $c = 0$. Otherwise, i is privileged by the Extension Rule 2. If $nx_i \neq 0$, then node i is adjacent to nodes nx_i and j. Since node i is not privileged by the Clean Rule 2, $cnbr^-(i)$ is not empty. Since a node of $c = 0$ and the node nx_i cannot be in $cnbr^-(i)$, node i has degree at least 3, a post. ■

Lemma 11: Upon stabilization, if there is an applicant or an f-post i with $c_i = 0$ and $|N_0(i)| = 0$, then the network does not have a Popular Matching.

Proof:

If i is an applicant, let p be a neighboring post. By Lemma 7, if c_p is odd, the network does not have a Popular Matching. Hence c_p is even, and $c_p \geq 2$. By Lemma 4, there is a matched path from p to an applicant of $c = 1$. Every node is matched in the matched path. Each applicant has at most one alternative post to match. If we force a match of i and p, another applicant along the matched path becomes unmatched. An applicant-complete matching does not exist in the network. By Lemma 2, there is no Popular Matching in the network.

Let i be an f-post and $a \in N(i)$. By Lemma 7, if c_a is odd, the network does not admit a Popular Matching. Hence c_a is even. If $nx_a \neq i$, then $|cnbr^0(i)| > 0$. By Lemma 10, node a is a post, a contradiction. Hence, $nx_a = i$. Since $c_i \neq 1$, i has at least 2 neighboring applicants. Let b be another neighbor of i, $nx_b = nx_a = i$. Since $c_i = 0$, a and b cannot be in the same matched path, $r_b \neq r_a$. By symetry, every neighboring post has a unique r. For proposition, let a have the largest r. Since nodes i and b are not in the matched path of a, $c_a < n - 1$. By definition, $i \in ufp$ and $a = mpe(i)$. Node i is privileged to make a move by the f Rule, a contradiction. ■

Lemma 12: Upon stabilization, if a node i with $c_i > 1$ and c_i is even has $|cnbr^0(i)| > 0$, then the network does not have a Popular Matching.

Proof:

By Lemma 10, node i is a post adjacent to at least 3 applicants. Let applicant j be a neighbor in $cnbr^0(i)$. Since $c_j \neq 1$, applicant j has two neighboring posts by the Start Rule. Let k be the other neighbor of j. By Lemma 7, if a node of $c = 0$ is adjacent to a node of an odd counter c, the network does not have a Popular Matching. Hence c_k is even. Let S_0 be the maximally connected component, connected to node j, on nodes of $c = 0$. For proposition, let S_0 exclude j. There are two cases.

Case one, every node in S_0 has at least two neighbors in S_0. Hence there is at least one cycle in S_0. Because the reduced graph G' is bipartite, every cycle is even. Let C be a cycle in S_0. The number of posts is equal to the number of applicants in C. Since an applicant has degree at most 2, cycle C must connect to j or with the rest of S_0 via one or more post(s) of degree at least 3. Since j is an applicant, the total number of posts is no more than the number of applicants in S_0. Note that if a node in S_0 is adjacent to a neighbor with $c > 0$, then the node has degree at least 3, a post. If every applicant is matched in S_0, then node j must match to node i which is not in S_0. By Lemma 4, there is a matched path from post i to an applicant of $c = 1$. Each matched applicant has at most one alternative post to match. If we force a match of i and j, another applicant along the matched path becomes unmatched. An applicant-complete matching does not exist in the network. By Lemma 2, there is no Popular Matching in the network.

Case two, there is at least one node l in S_0 such that l has only one neighbor in S_0. Let $s \in N(l)$ be the unique neighbor in S_0. Since $c_l \neq 1$, l has at least two neighbors by the Start Rule. Let t with $c_t > 0$ be another neighbor of j. By Lemma 7, c_t is even otherwise the network does not have a Popular Matching. By Lemma 5, t is matched with a node in $cnbr^-(t)$. Since l is not privileged by the Extension Rule 3, $l \notin inm$ or $mpe(l) = null$. Every nonzero neighbor of l has an even counter c. If $l \notin inm$, then there exists a neighbor with $nx \neq l$. For proposition, let t be the neighbor. By Lemma 10, t is a post. Similar to node i, by Lemma 4, there is a matched path from post t to an applicant of $c = 1$. Each matched applicant has at most one alternative post to match. Any path between applicants j and l has more applicants than posts. If we force a match of i and j, or a match of t and l, another applicant along the matched path becomes unmatched. An applicant-complete matching does not exist in the network. By Lemma 2, there is no Popular Matching in the network, a contradiction. Hence $l \in inm$ and $nx_t = l$. By symmetry, every neighbor with $c > 1$ has $nx = l$. Since $c_l = 0$, every neighbor is in a separate matched path with a unique r. For proposition, let t have the largest r. Since nodes i and l are not in the matched path of t, $c_t < n - 1$. By definition, $t = mpe(l)$. Node t is privileged to make a move by the Extension Rule 3, a contradiction. ■

Lemma 13: If a network admits a Popular Matching, a node i with $c_i = 0$ may not have exactly one neighbor of $c = 0$ upon stabilization.

Proof:

Proof by contraction: let i be a node with $c_i = 0$. By the Start Rule, node i has degree at least 2. Upon stabilization, let $o \in N(i)$ be the sole neighbor with $c_o = 0$. Every neighbor of i, except for o, has $c > 1$. Let $j \in N(i)$ and $c_j \geq 1$. Since the network admits a Popular Matching, c_j is even by Lemma 7, and $nx_j = i$ by Lemma 12. Hence, node i is in inm. Since i is not privileged by the Extension Rule 3, $mpe(i)$ must be $null$. Since $c_i = 0$, every neighbor of i is in a separate matched path with a unique r. For proposition, let j have the largest r. Since nodes i and o are not in the matched path of j, $c_j < n - 1$. By definition, $j \in mpe(t)$, a contradiction. ■

Lemma 14: Let the network admit a Popular Matching. Upon stabilization, if a node i has $c_i = 0$ and $|N_0(i)| = 0$, then it is a non-f-post and the only node with $c = 0$ in the network.

Proof:

By Lemma 11, i is a non-f-post. Let a be an applicant adjacent to i. By Lemma 7, c_a is even. Otherwise, the network does not admit a Popular Matching. Since $c_a > 0$, c_a is at least 2. By Lemma 4, there is a matched path from a to a post of degree 1. Every post has an odd counter c in the matched path of a. Because each applicant has degree at most 2 in the reduced graph G', the path may only branch at one or more posts. Let p be a post in the matched path with a neighboring

applicant b which is not in the same matched path, $r_b \neq r_p$. Since b is not privileged by the Clean Rule 2,$cnbr^-(b) \neq \emptyset$ and $p \notin cnbr^-(b)$. If $c_b \leq 1$, then p is privileged by the Clean Rule 3. Hence $c_b > 1$. If c_b is odd, applicant b is active by either the Clean Rule 2 or the Clean Rule 4. Hence c_b is even. Every post has an odd counter c in the matched path of b. This is the same situation as the matched path of a. By the same argument in any further branches, there is no node of $c = 0$. ■

Lemma 15: Let the network admit a Popular Matching. Upon stabilization, if there is an unmatched node with $c > 0$, then it is a non-f-post with $c = 1$ and there is no node of $c = 0$ in the network.

Proof:

Let i be an unmatched node with $c_i > 0$. By Lemmas 5 and 8, every node with $c > 1$ is matched. Hence $c_i = 1$. By the Start Rule, i has degree 1, $r_i = i > 0$ and $nx_i = u(i)$. Hence $i \in ps(u(i))$. Since node i is not matched, $u(i)$ cannot be in the matched path of i, $r_{u(i)} \neq r_i$. Therefore, node i is not an f-post. Otherwise, $u(i)$ is privileged by the Extension Rule 1. If $c_{u(i)} = 0$, since $u(i)$ is not privileged by the Extention Rule 1, $pp(u(i)) = null$. Since $i \in ps(u(i))$, $u(i)$ must be a post with $|ps(u(i))| > 1$. By Lemma 6, there is no Popular Matching in the network, a contradiction. Note that the network is connected with at least 4 nodes, hence $c_{u(i)} > 1$. If $c_{u(i)}$ is odd and $nx_{u(i)} \neq i$, then $|cnbr^1(u(i))| > 0$. Node $u(i)$ is privileged by the Clean Rule 3, a contradiction. If $c_{u(i)}$ is odd and $nx_{u(i)} = i$, then $u(i)$ is privileged by the Clean Rule 4, a contradiction. Hence, $c_{u(i)}$ is even. By Lemma 5, $u(i)$ is matched with a node in $cnbr^-(u(i))$. Any node in $cnbr^-(u(i))$ is also in $ps(u(i))$. Since $r_{u(i)} \neq r_i$, $i \notin cnbr^-(u(i))$. If node i is an applicant, then $u(i)$ is a post with $|ps(u(i))| > 1$. The network does not have a Popular Matching, a contradiction. Hence, i is a non-f-post and $u(i)$ is an applicant.

Since $c_{u(i)} \geq 2$, by Lemma 4, there is a matched path from $u(i)$ to a post of degree 1. Every post has an odd counter c in the matched path. Because each applicant has degree at most 2 in the reduced graph G', the path may only branch at one or more posts. Let p be a post in the matched path with a neighboring applicant b which is not in the same matched path, $r_b \neq r_p$. Since b is not privileged by the Clean Rule 2, $cnbr^-(b) \neq \emptyset$ and $p \notin cnbr^-(b)$. If $c_b \leq 1$, then p is privileged by the Clean Rule 3. Hence $c_b > 1$. If c_b is odd, applicant b is active by either the Clean Rule 2 or the Clean Rule 4. Hence c_b is even. Every post has an odd counter c in the matched path of b. This is the same situation as the matched path of $u(i)$. By the same argument in any further branches, there is no node of $c = 0$. ■

Lemma 16: Let the network admit a Popular Matching. Upon stabilization of Algorithm MPM, there are three possible outcomes: (1) There is no node with $c = 0$; (2) There is one non-f-post with $c = 0$; (3) There are more than one node with $c = 0$. Each node, call it i, with $c_i = 0$ has $|N_0(i)| = 2$ and they form one or more disjoint even cycles.

Proof:

Upon stabilization, a network can have either (1) no node with $c = 0$, or (2) and (3) some node(s) with $c = 0$. For (2), by Lemma 14, if a node i has $c_i = 0$ and $|N_0(i)| = 0$, then it is a non-f-post and the only node with $c = 0$. For (3), by Lemma 13, a node i with $c_i = 0$ may not have exactly one neighbor in $N_0(i)$ upon stabilization. Therefore, if there are more than one nodes of $c = 0$, then every node i of $c_i = 0$ have $|N_0(i)| \geq 2$.

Since the network admits a Popular Matching, we have $|P| \geq |A|$ by Halls Marriage Theorem [9]. By Lemma 15, if there is an unmatched node with $c > 0$, then there is no node of $c = 0$ in the network. Thus in the network with nodes of $c = 0$, every nodes of $c > 0$ is matched upon stabilization. The numbers of matched applicants and posts with $c > 0$ are the same. Let P_0 and A_0 denote the posts and applicants with $c = 0$. We have $|P_0| \geq |A_0|$. The reduced graph G' is bipartite. All the edges are between applicants and posts. Let m_0 denote the number of edges between P_0 and A_0. Since every node i of $c_i = 0$ has $|N_0(i)| \geq 2$, we have $2|P_0| \leq m_0$. However, each applicant may have at most two edges, thus $m_0 \leq 2|A_0|$. We have $2|P_0| \leq 2|A_0|$ or $|P_0| \leq |A_0|$. Therefore $|P_0| = |A_0|$. Every post with $c = 0$ has exactly two neighboring applicants of $c = 0$. The graph induced by all the nodes of $c = 0$ is a family of disjoint even cycles. ■

A. Use Algorithm $MMEC$ with Algorithm MPM

By Lemma 16, MPM may generate one or more disjoint even cycles of nodes with $c = 0$. We apply $MMEC$ on nodes with $c = 0$ and $|N_0(i)| = 2$. Corollary 1 follows directly from Lemmas 1 and 16.

Corollary 1: If the network admits a Popular Matching, every node with $c = 0$ and $|N_0| = 2$ is matched upon stabilization.

Lemma 17: If a network admits a Popular Matching, the proposed algorithms produce an applicant complete matching.

Proof:

Assume the network admits a Popular Matching. By Lemma 9, every applicant with $c > 0$ is matched. By Lemma 16, every applicant with $c = 0$ is in an even cycle of nodes with $c = 0$. By Corollary 1, every applicant with $c = 0$ is matched upon stabilization. ■

Lemma 18: If a network admits a Popular Matching, every f-post is matched.

Proof:

Assume the network admits a Popular Matching. By Lemma 9, every f-post with $c > 0$ is matched. By Lemma 16, every f-post with $c = 0$ is in an even cycle of nodes with $c = 0$. By Corollary 1, every f-post with $c = 0$ is matched upon stabilization. ■

Lemma 19: If a network admits a Popular Matching, Algorithms MPM and $MMEC$ stabilize at one of maximum cardinality.

Proof:

Assume a Popular Matching exists. By Lemma 17, every applicant is matched. The proposed algorithms produce an applicant-complete matching. This is the maximum cardinality possible. ■

Theorem 1: If a Popular Matching exists in the network, self-stabilizing algorithms MPM and $MMEC$ produce a Maximum Popular Matching in a stable configuration.

Proof:

By combining Lemmas 2, 17, 18 and 19. Lemma 17 proves condition (ii) of Lemma 2. Lemma 18 proves condition (i) of Lemma 2. Hence self-stabilizing algorithms MPM and $MMEC$ produce a Popular Matching if one exists. Lemma 19 shows that the resulting matching has maximum cardinality. ■

V. Complexity Analysis

In this section, we establish the complexity of our algorithms. We define the concept of a *Cause Node*.

Definition. *A node i is a* cause node *if and only if (1) it has moved by the Start Rule; or (2) it has never moved since the start of the self-stabilizing algorithm.*

Lemma 20: If a node moves by the Start Rule of Algorithm MPM, it is the only move of the node in the self-stabilizing process.

Proof:

The predicate of the Start Rule requires a node has degree one in the reduced graph G'. The rules are executed in list order. Once the node moves by the Start Rule, its counter $c = 1$. It prevents the node from moving again. To be privileged for the Clean Rules, either the node has degree greater than 1 or its counter $c > 1$. The predicates of the Extension Rules require $c = 0$ or $c > 1$. The f Rule requires $i \in ufp$ which requires $c = 0$. In Algorithm $MMEC$, all rules require $c = 0$. ■

Lemma 21: There are no more than n moves by the cause nodes.

Proof:

After a move by a cause node of types (2), it may become a cause node of type (1) or it may no longer be a cause node. By Lemma 20, a cause node of type (1) cannot move again. ■

Lemma 22: Only a cause node of type (2) can be privileged by the Clean Rule 1.

Proof:

The Start Rule is the only rule that sets $c = 1$. It requires $|N(i)| = 1$ and $c_i \neq 1 \vee r_i \neq i \vee nx_i \neq u(i)$. After a move by the Start Rule, $c_i = 1$, $r_i = i$ and $nx_i = u(i)$. To be privileged by the Clean Rule 1, node i must have $c_i = 1$ and $|N(i)| > 1$. Therefore, node i has never moved. It is a cause node of type (ii), and it cannot be privileged by the Clean Rule 1 again. ■

Corollary 2: There are no more than n moves by the Clean Rule 1.

Lemma 23: If a node has moved and it is privileged by the Clean Rule 3, then the node has degree at least 3, a post.

Proof:

By the Clean Rule 3, c_i is odd and $|cnbr^0(i)| + |cnbr^1(i)| > 0$. The Extension Rule 3 and the f Rule are the only rules that set counter c to an odd number. Therefore, the last move of i is by the Extension Rule 3 or the f Rule. A move by either rules sets nx_i to a neighbor and results in $|cnbr^0(i)| + |cnbr^1(i)| = 0$. Hence, at least one neighbor $j \neq nx_i$ has set counter c_j to 0 or 1 before i is privileged by the Clean Rule 3. Because node i is not privileged by the Clean Rule 2, $cnbr^-(i) \neq \emptyset$. Therefore, node i has degree at least 3. An applicant has degree at most 2 in the reduced graph G'. Node i must be a post. ■

Lemma 24: A node i is privileged by the Clean Rule 2 only if:
(i) node i is a cause node of type (2), or
(ii) a neighbor $j \in cnbr^-(i)$ was a cause node of type (2) and j has moved before i is privileged by the Clean Rule 2, or
(iii) node j has moved by the Clean Rules 2 or 3.

Proof:

Let i be the node privileged by the Clean Rule 2. Assume node i has moved, otherwise it is a cause node of type (2). By the predicate of the Clean Rule 2, $c_i > 1$ and $cnbr^-(i) = \emptyset$. Hence the last move of i is not by any of the Clean Rules and $cnbr^-(i) \neq \emptyset$ right after the move. Note that no rule can point nx_i to a node with $c > 0$. Hence, at least one node in $cnbr^-(i)$ has moved before node i is privileged by the Clean Rule 2. Let j be a neighbor that later would move out of $cnbr^-(i)$.

Since the rules are tried in order, our proof follows the list. If j has made its first move and/or it has moved by the Start Rule, or the Clean Rule 1, then it was a cause node of type (2) by definition, Lemmas 20 or 22. Otherwise, $c_j > 1$. If j has moved by the Clean Rules 2 or 3, the lemma stands true. After the move of i by the Extension Rules 1, 3 or the f Rule, $r_i = r_j$ and $c_i = c_j + 1$. Node j cannot move by the Clean Rule 4.

Next, assume node j is not privileged by any of the Clean Rules. Because $r_j = r_i$ and $c_j > 1 \neq 0$, j cannot move by the Extension Rule 1. Because $nx_j = i$, j cannot move by the Extension Rule 2. Since $c_j \neq 0$, j cannot move by the Extension Rule 3 or the f Rule. ■

Lemma 25: A node i is privileged by the Clean Rule 3 only if:
(i) node i is a cause node of type (2), or
(ii) a neighbor $j \in N(i)$ was a cause node of type (2) and j has moved before i is privileged by the Clean Rule 3, or
(iii) a neighbor has moved by the Clean Rule 2.

Proof:

Let i be the node privileged by the Clean Rule 3. Assume node i has moved, otherwise it is a cause node of type (2). By the predicate of the Clean Rule 3, $c_i > 1$ and c_i is odd. In Algorithm MPM, two rules can move c to an odd value greater than 1. They are the Extension Rule 3 and the f Rule. Therefore, the last move of i is by one of the two rules. A move by either rule results in $cnbr^0(i) = cnbr^1(i) = \emptyset$. Therefore, a neighbor $j \neq nx_i$ has moved into $cnbr^0(i)$ or $cnbr^1(i)$ before i is privileged by the Clean Rule 3. By Lemma 23, node i is a post. Hence, j is an applicant.

The Start Rule is the only rule that sets counter c to 1. The four Clean Rules are the only rules that set $c = 0$. Since the rules are tried in order, our proof follows the list. If j has made its first move and/or it has moved by the Start Rule, or the Clean Rule 1, it was a cause node of type (2) by definition, Lemmas 20 or 22. Otherwise, j has moved before it moves

into $cnbr^0(i)$ or $cnbr^1(i)$. If node j moves into $cnbr^0(i)$ by the Clean Rule 2, the lemma stands true. Applicant j cannot move by the Clean Rule 3 by Lemma 23. If j is privileged by the Clean Rule 4, then $c_j > 1$ and c_j is odd. Hence $i \notin inm$ and $i \notin ufp$. Node i could not have moved by the Extension Rule 3 or the f Rule, a contradiction. ■

Lemma 26: A node i is privileged by the Clean Rules 2 or 3 only if:
(i) node i is a cause node of type (2), or
(ii) a cause node of type (2) has moved a finite number of rounds before.

Proof:

By Lemma 24, a node is privileged by the Clean Rule 2 only if the node is a cause node of type (2), or a moved neighbor was a cause node of type (2), or a neighbor has moved by the Clean Rules 2 or 3. By Lemma 25, a node is privileged by the Clean Rule 3 only if the node is a cause node of type (2), or a moved neighbor was a cause node of type (2), or a neighbor has moved by the Clean Rule 2. There is a finite number of moves since the start of algorithm MPM. The first move of any node is a move by a cause node of type (2) by definition. ■

Lemma 27: A move by a cause node of type (2) initiates no more than $2n^2$ moves of the Clean Rule 2, and no more than $2n$ moves of the Clean Rule 3.

Proof:

Let i be a cause node of type (2). Let $j = nx_i$ be a neighbor of node i which is in $cnbr^-(j)$. After i moves out of $cnbr^-(j)$, j may become privileged by the Clean Rule 2. If node j moves, a neighbor $k = nx_j \neq i$ with $j \in cnbr^-(k)$ may become privileged by the Clean Rule 2, and so on. Since $c \leq n$ in algorithm MPM, there are no more than n moves by the Clean Rule 2 on nodes of $r = r_i$. Let $C2_i$ denote the ID sequence of all the nodes moved consecutively by the Clean Rule 2 with $r = r_i$. If jk is in $C2_i$, then $j \in cnbr^-(k)$ before the move of j by the Clean Rule 2. Note that the next ID after jk cannot be j, because $nx_k \neq j$ by the definition of $cnbr^-(k)$. Each node has up to 2 neighbors in sequence $C2_i$. If a node has 2 neighbors in $C2_i$, the two neighbors are different nodes.

Let l denote a node in $C2_i$. By Lemma 25, a node $s \in N(l)$ may become privileged by the Clean Rule 3 after node l moves by the Clean Rule 2. If s has never moved before, we contribute further moves to the cause node s. Hence, assume s has moved. By Lemma 23, node s is a post of degree at least 3, and l is an applicant. Node l has degree at most two and one of the neighbor is s. By the Clean Rule 3, $l \in cnbr^0(s)$ and $nx_s \neq l$. Hence $s \notin cnbr^-(l)$. Since node s is not privileged by the Clean Rule 2, $l \notin cnbr^-(s)$. There are at most two applicants in $C2_i$ that can possibly satisfy the requirements for l: node i, because $cnbr^-(i)$ can be empty; and the node of the largest c in $C2_i$. A node of the largest c is never in $cnbr^-$ of any node in $C2_i$. Since s is privileged by the Clean Rule 3, $c_s > 1$ and c_s is odd. The Extension Rule 3 and the f Rule are the only rules in algorithm MPM that set a counter c to an odd value. Therefore, the last move of node s, before the move of l by the Clean Rule 2, is by the Extension Rule 3 or the f Rule. Either rule sets $r_s = r_{mpe(s)}$ and requires $s \in inm$ or $s \in ufp$. By the definitions of inm and ufp, c_l is even and $nx_l = s$. Since the move of l by the Clean Rule 2 moves l into $cnbr^0(s)$, $c_l \neq 0$. After the move of l by the Clean Rule 2, s is not privileged by the Clean Rule 2, hence $l \notin cnbr^-(s)$. By the definition of $mpe(s)$, $r_s > r_l$ after the last move of s by the Extension Rule 3 or the f Rule. Before node s moves by the Clean Rule 3, l may have moved several times by the Clean Rule 2. To establish the upper bound of moves, we attribute the moves to the move of l by the Clean Rule 2 immediately before the move of s by the Extension Rule 3 or the f Rule. For proposition, let $r_l = r_i$ when l is privileged by the Clean Rule 2. Hence $r_s > r_l = r_i$. Every move by the Clean Rule 3 results in a larger r. Since $r \leq n$ in algorithm MPM, the total number of moves of the Clean Rule 3, caused by i, is less than $2n$.

Now look at $C2_s$. Because s is a post, there is at most one node in $C2_s$ that can cause a neighbor to move by the Clean Rule 3—the node of the largest c in $C2_s$. In between two moves of the Clean Rule 3, there are at most n moves (one $C2$ sequence) by the Clean Rule 2. Hence one cause node initiates less than $2n^2$ moves by the Clean Rule 2. ■

Lemma 28: There are $O(n^3)$ moves by the Clean Rules 2 and 3.

Proof:

By Lemma 26, a move by the Clean Rule 2 or 3 necessitates a move of a cause node of type (2). By Lemma 27, a move of a cause node of type (2) can cause $O(n^2)$ moves by the Clean Rules 2 and 3. After one move, a node is no longer a cause node of type (2). Hence, the total number of moves by the Clean Rules 2 and 3 is $nO(n^2) = O(n^3)$. ■

Lemma 29: There are $O(n^3)$ moves by the Clean Rule 4.

Proof:

Let i be a node that is privileged by the Clean Rule 4. Hence $c_i > 1$ and c_i is odd. If i has never moved before, then it is a cause node of type (2). By Lemma 21, the total number of moves by cause nodes is no more than n. Assume node i has moved before. In Algorithm MPM, two rules can move c to an odd value greater than 1. They are the Extension Rule 3 and the f Rule. The last move of i must be one of the two rules. A move of either rule sets $r_i = r_{mpe(i)}$, $c_i = c_{mpe(i)} + 1$ and nx_i points to a neighbor, call it j. Hence $i \in ps(j)$.

Case 1: the last move of i is by the Extension Rule 3. Hence c_j is 0 before and immediately after the move. Because c_i is odd, $j \notin inm$ and $j \notin ufp$. Hence j's last move which sets $c_j \neq 0$ is by the Extension Rule 1. Since i is privileged by the Clean Rule 4, $r_j \neq r_i$ or $c_j \neq c_i + 1$. By the definition of $pp(j)$, this is possible only when j is an applicant and i is $s(j)$. Since the last move of i is by the Extension Rule 3, $i \in inm$. By the definition of inm, every neighboring applicant of i has an even counter c and $c \neq 0$ after node j moves by the Extension Rule 1. For i to be privileged by the Clean Rule 4 again, one of i's neighboring applicants must move by the Clean Rules 1 or 2. They are not privileged by the Clean Rules 3 or 4 because their c's are even.

Case 2: the last move of i is by the f Rule. Hence $i \in ufp$ before the move and $i \in pufp$ immediately after the move.

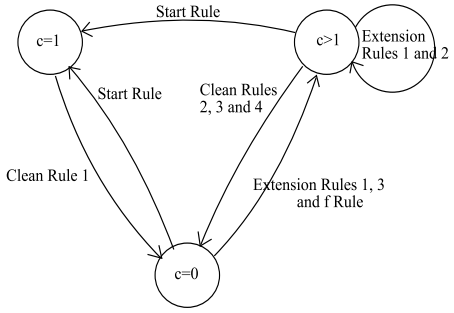

Fig. 1. State-transitions of Algorithm MPM

Every neighboring applicant has an even counter c and $c \neq 0$. For i to become privileged by the Clean Rule 4, it must move out of $pufp$. One of the neighboring applicant must move by the Clean Rules 1 or 2. They are not privileged by the Clean Rules 3 or 4 because their c's are even.

For both cases 1 and 2, there are no more than n moves by the Clean Rule 1 by Corollary 2. There are $O(n^3)$ moves by the Clean Rule 2 by Lemma 28. Since each neighboring applicant has at most two neighbors. The total number of moves by the Clean Rule 4 is of $O(n)+2O(n^3) = O(n^3)$. ■

Lemma 30: There are no more than $O(n^3)$ moves by the Extension Rule 3 and the f Rule.

Proof:

Figure 1 is the diagram of the state transitions of counter c. By Lemmas 28 and 29, there are $O(n^3)$ moves by the Clean Rules 2, 3 and 4. Every move by the Extension Rule 3 or the f Rule changes the state from $c = 0$ to $c > 0$. Hence the total number of moves by the Extension Rule 3 and the f Rule is $O(n^3)$. ■

Theorem 2: There is a total of $O(n^3)$ moves by the Algorithm MPM.

Proof:

Figure 1 is the diagram of the state transitions of counter c. By Lemma 20 and Corollary 2, there are $O(n)$ moves between the states of $c = 1$ and $c = 0$, and between the states of $c = 1$ and $c > 1$. By Lemmas 28 and 29, there are $O(n^3)$ moves between the states of $c > 1$ and $c = 0$. Hence the number of moves that change the nodes' states is of $O(n^3)$. Next, we bound the number of moves that do not change the state of counter c.

Let node i with $c_i > 1$ move by the Extension Rule 1. Hence, i is an applicant. Each such move requires either i is a cause node of type (2), or $f(i)$ moves by the f Rule and $i = nx_{f(i)}$. By Lemma 21, there are $O(n)$ moves by the cause nodes. By Lemma 30, there are $O(n^3)$ moves by the f Rule. Hence, the number of moves by the Extension Rule 1 that do not change the state of counter c is $O(n^3)$.

Let node i be privileged by the Extension Rule 2. Hence c_i is even. If i has never moved before, then it is a cause node of type (2). Assume node i has moved before. In Algorithm MPM, only the Extension Rule 1 can move c to an even value greater than 1. The last move of i must be by the Extension Rule 1. After node i moves by the Extension Rule 2, $nx_i \neq 0$. Hence, i cannot move again by the Extension Rule 2 before it makes another move by the Extension Rule 1 in a later round. Therefore, the total number of moves by the Extension Rule 2 is no more than the total number of moves by the Extension Rule 1, $O(n^3)$. ■

A. Complexity of Algorithm $MMEC$

Algorithm $MMEC$ executes on nodes with $c = 0$ and $|N_0(i)| = 2$. Let G'' be the induced subgraph of G' on these nodes. Since each node has degree 2, network G'' is one or more disjoint paths and/or even cycles. We further require that Algorithm MPM sets $c0$, $r0$ and $nx0$ to 0 when it moves a node's counter c to 0.

Lemma 31: Between 2 moves of Algorithm MPM, there are $O(n^2)$ moves by Algorithm $MMEC$.

Proof:

The Start Rule is the only rule of Algorithm $MMEC$ that sets $c0 = 1$. If a node i with $c0_i = 1$ moves by the Start Rule, its $r0_i = i$ and $nx0_i = x_i$. The Clean Rules 1 and 2 are the only rules of $MMEC$ that set $c0 = 0$. Both rules ensure $r0 = nx0 = 0$. When Algorithm MPM moves any node to join G'', its $c0$, $r0$ and $nx0$ are set to 0. Hence if a node moves by the Clean Rule 1 of $MMEC$, it has never moved by MPM or $MMEC$ before. The total number of such cases is n. Hence, the Clean Rule 1 of $MMEC$ makes $O(n)$ moves during the entire self-stabilizing process.

Next, let i with $c0_i > 1$ move by the Clean Rule 2 of $MMEC$. The Extension Rule of $MMEC$ is the only rule that creates nodes of $c0 > 1$. If node i has moved, the last move must be the Extension Rule. Immediately after the move, $cnbr0^-(i) \neq \emptyset$. Let node j be in $cnbr0^-(i)$. Notice that node i is not privileged by the Extension Rule. Node j must move before i is privileged by the Clean Rule 2. If j has moved before, j can be privileged only by the Clean Rule 2. Both nodes i and j were in the same matched path before they move by the Clean Rule 2. Hence we can trace a sequence of moves by the Clean Rule 2 in the matched path of node i. Graph G'' is a collection of one or more disjoint paths and/or even cycles. In the connected path or cycle, the node at the lower end of the matched path of i has (1) moved for the first time, and/or (2) or adjacent to a node moved by Algorithm MPM. Between 2 moves of Algorithm MPM, the number of moves by the Clean Rule 2 tracing to case (1) is of $O(n^2)$. The number of moves by the Clean Rule 2 tracing to case (2) is of $O(n)$.

If a node moves by the Start Rule a second time, either at least one neighbor has moved by one of the Clean Rules of $MMEC$, and/or i is adjacent to a node moved by MPM. Each node incidents at most 2 nodes in G''. Hence, the Clean Rule 1 of $MMEC$ leads to at most $O(n)$ moves by the Start Rule. Because the Start Rule requires $i > r0_{x/y}$, each move by the Start Rule occurs at the higher end of its matched path. Hence, the Clean Rule 2 tracing to case (1) leads to $O(n)$ moves by the Start Rule. Moves by the Clean Rule 2 tracing to case (2) leads to a constant number of moves by the Start Rule.

Let node i move by the Extension Rule. Hence $mpe0(i) \neq null$. Let node j be $mpe0(i)$. If j has moved before, its last move must be the Extension Rule or the Start Rule.

We can trace a sequence of moves by the Extension Rule in the matched path of node i. In the connected path or cycle, either (3) the node at the lower end of the matched path has never moved, or (4) it is adjacent to a node moved by MPM, or (5) it has moved by the Start Rule. Between 2 moves by Algorithm MPM, moves by the Extension Rule tracing to case (3) is $O(n^2)$. For case (4), there is a linear number of moves tracing to an end of the path of G''. Since each Start Rule initiates one matched path, there are $O(n^2)$ moves by the Extension Rule of case (5). ■

Theorem 3: Algorithms MPM and $MMEC$ stabilize in $O(n^5)$ moves.

Proof:

By Theorem 2, there is a total of $O(n^3)$ moves by Algorithm MPM. By Lemma 31, there are $O(n^2)$ moves by Algorithm $MMEC$ between 2 moves of MPM, hence the result. ■

REFERENCES

[1] D. J. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn. Popular matchings. *SIAM J. Comput.*, 37(4):1030–1045, 2007.
[2] J. Beauquier, A. K. Datta, M. Gradinariu, and F. Magniette. Self-stabilizing local mutual exclusion and daemon refinement. In *International Symposium on Distributed Computing*, pages 223–237, 2000.
[3] E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Comm. ACM*, 17(11):643–644, Jan. 1974.
[4] S. Dolev. *Self-Stabilization*. MIT Press, 2000.
[5] S. Dolev, A. Israeli, and S. Moran. Uniform dynamic self-stabilizing leader election. *IEEE Trans. on Parallel and Distributed Systems*, 8(4), 1995.
[6] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.
[7] P. Gardenfors. Match making: assignments based on bilateral preferences. *Behavioral Sciences*, 20:166–173, 1975.
[8] D. Gusfield and R. W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, 1989.
[9] P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, 10:26–30, 1935.
[10] S. M. Hedetniemi, S. T. Hedetniemi, D. P. Jacobs, and P. K. Srimani. Self-stabilizing algorithms for minimal dominating sets and maximal independent sets. *Comput. Math. Appl.*, 46(5-6):805–811, 2003.
[11] S.-C. Hsu and S.-T. Huang. A self-stabilizing algorithm for maximal matching. *Inform. Process. Lett.*, 43:77–81, 1992.
[12] A. Panconesi and A. Srinivasan. The local nature of δ-coloring and its algorithmic applications. *Combinatorica*, 15:225–280, 1995.
[13] S. Rajagopalan and V. Vazirani. Primal-dual RNC approximation algorithms for set cover and covering integer programs. *SIAM J. Comput.*, 28:525–540, 1998.
[14] Z. Shi, W. Goddard, and S. T. Hedetniemi. an anonymous self-stabilizing algorithm for 1-maximal independent set in trees. *Information Processing Letters*, 91(2):77–83, 2004.
[15] G. Tel. *Introduction to Distributed Algorithms, Second Edition*. Cambridge University Press, Cambridge UK, 2000.