Stabilizing Unstable Matching because of a Varied Preference List in Distributed Stable Marriage Problem

Hideki Kinjo†, Morikazu Nakamura†, and Kenji Onaga‡

†Department of Information Engineering, University of the Ryukyus
1 Senbaru, Nishihara, Okinawa, 903-0213 JAPAN
‡Okinawa Research Center, Telecommunication Advancement Organization
1 Asahi-machi, Naha, Okinawa 900-0029, Japan
†Phone: +81-98-895-8715, †Fax: +81-98-895-8727,
E-mail: kin@ads.ie.u-ryukyu.ac.jp

Abstract: A distributed stable marriage problem in extended in this paper. The traditional approach related to the stable matching problem assume that preference lists are fixed. However, in decentralized version this assumption is not reasonable because of the autonomy of members.
In this paper, we consider the situation that a preference list can be varied at some stable matching and show the condition to be broken the stability of the original matching. Moreover, we propose a stabilization algorithm to obtain a stable matching by execution with a minimal set of members.

1. Introduction

The original stable marriage problem was proposed by D. Gale and L. Shapley [1]. This problem is applicable to wide area of the real world. Many researchers have treated this problem in different ways. Some of the results are surveyed in the book [2, 3]. In [4], the gender-fair stable marriage problem was shown as NP-hard and an approximate solution was proposed.
In [5], we proposed a distributed stable marriage problem in which each member is regarded as a distributed element. By message exchanging, they communicate with each other and obtain a partner for a stable marriage. The distributed version of the problem is applicable to cooperative works of autonomous robots [5, 6].
In [7], we proposed a distributed algorithm, called GS-algorithm, and introduced an autonomous mechanism for exchanging partners. A matching reconstruction (in [5], called divorce process) is to exchange partners which is initiated by some member who has complaint about the matching partner by breaking the current matching to get a preferable partner. This break will initiate message communication process based on the GS-algorithm to generate another matching.
In previous works related on the stable matching problem, the PLs are assumed to be fixed [1, 2, 3, 5, 7]. However, once we extend the original problem into the autonomous distributed version, we need to consider the situation that the preference list can be varied. In this paper, we allow preference lists to be varied at a stable matching. In this case, the stability may be broken. Therefore, we propose a stabilizing algorithm to obtain another stable matching by execution to which a minimal set of members join, while all the members should join when we employ the GS-algorithm[3].
This paper is organized as follows: In Section 2, we give some notations and definitions for the distributed stable marriage problem considered in paper and we show GS-algorithm and matching reconstruction. In Section3, we present a stabilizing algorithm for varied preference list. Section 4 concludes this paper.

2. Distributed Stable Marriage Problem

In this section, we give the definitions of the original stable marriage problem and the distributed stable marriage problem and describe an algorithm to obtain a stable partner, called GS-algorithm.
The basic problem [1] is explained as follows: We consider two sets of equal size n, the man group and the woman group,

$$\boldsymbol{m} = \{m_1, m_2, \cdots, m_n\},$$
$$\boldsymbol{w} = \{w_1, w_2, \cdots, w_n\}.$$

Each member in both sets possesses a strictly ordered preference list PL, containing all the members of the opposite sex. Let p be a person (man or woman) and q, r be members in the opposite sex of p. A person p prefers q to r if q precedes r on p's preference list. Two pairs (m_i, w_i) and (m_j, w_j) are said "unstable" if conditions

$$m_i \text{ prefers } w_j \text{ to } w_i, \text{ and}$$
$$w_j \text{ prefers } m_i \text{ to } m_j$$

hold. When unstable, there is a high probability of break-up and a new pair formation (m_i, w_j). A matching $\boldsymbol{M} = \{(m_1, w_1), (m_2, w_2), \cdots, (m_n, w_n)\}$ is said "stable" if no two pairs in $\boldsymbol{M}$ are unstable.

For an instance of the stable marriage problem, there exist generally many stable matchings, in the worst case exponential order [2]. In [2], it is shown that the set of all stable matchings forms a distributive lattice under a natural ordering relation.

Definition 1 *Stable matching M is said to dominate stable matching M', represented by $M \preceq M'$, if every man either prefers the partner in M to in M' or has the same partner .*

Theorem 1 *[2] For a given instance of the stable marriage problem, the partial order $(M, \preceq)$ forms a distributive lattice L, with the meet and join of M and M' denoted by $M \wedge M'$ and $M \vee M'$, where $M \wedge M'$ ($M \vee M'$) is the stable matching in which each man receives the better (poor) of his partners in M and M'.*

The minimum and the maximum of the lattice represent the man-optimal matching and the woman-optimal matching, respectively.
The distributed version is defined as follows: There are two groups of equal size n in a distributed environment. They can communicate with each other by message passing. We assume that any communication delay and computation time are finite. The solution of the problem is to develop an algorithm that transforms from the following initial situation to the destination by the finite number of message communications and the computation in each processor.
[**Initial Situation**] Each member(robot) belongs to one of two groups, man group and woman group, knows the identifier number(ID) of all the others and their belonging group. Each member has his/her preference list.
[**Destination**] Each member knows his/her partner and the matching is stable. In this case, we say that each member has a stable partner.
We proposed a distributed algorithm, called Gale-Shapley based algorithm (or GS-algorithm) [7], that can get a stable matching on the distributed stable marriage problem. Some of the messages used in the proposed algorithm are defined as follows:

Propose: the sender proposes to the receiver,

Accept: the sender accepts the receiver's previous proposal,

Reject: the sender rejects the receiver's previous proposal,

Proposed: the sender informs the receiver that the sender is proposed.

The followings are the primitive commands for the message transmission in the proposed algorithm:

Send(p, Mes): to send a message Mes to p,

Receive(p, Mes): to receive a message Mes from p.

The GS-algorithm is shown in Figure 1. In this algorithm, man group's and woman group's members play different roles, proposer or receiver. When man group's members perform as proposer the GS-algorithm is called the man-optimal GS-algorithm, since obtained stable matching dominates the others. The algorithm reversed the roles is called the woman-optimal GS-algorithm.

```
1  function proposer(var i: integer): integer;
2  var w: integer;
3  begin
4   repeat
5    w := the i-th member of PL;
6    Send(w, Propose);
7    Receive(w, Mes);
8    case Mes of
9     'Reject': i := i + 1;
10    'Accept': break the loop;
11   end;
12  until;
13  proposer := w;
14 end.
15
16 function receiver(var m: integer): integer;
17 var p: integer;
18 begin
19  repeat
20   Receive(p, Mes);
21   case Mes of
22    'Propose':
23     if m := null then
24     m := p;
25     Send(every woman who participates,Proposed)
26     else
27      begin if prefer p to m then
28      Send(m, Reject);
29      m := p;
30      else
31      Send(p, Reject);
32      end;
33     end;
34    'Proposed':
35     if received Proposed from every woman then
36     Send(m, Accept); break the loop;
37     end;
38   end;
39  until;
40  receiver := m;
41 end.
42
43 procedure MAN;
44 var i, partner: integer;
45 begin i := 1;
46   partner := proposer(i);
47 end;
48
49 procedure WOMAN;
50 var m, partner: integer;
51 begin m := null;
52   partner := receiver(m);
53 end.
```

Figure 1. GS-algorithm

The members in the both groups basically want to obtain preferable partners. A member who got a worse partner may complain and be eager for a preferable partner. In this section we show an autonomous mechanism for exchanging partners (Figure 2). Here we define a new message for an autonomous mechanism as follows:

Divorce: the sender informs the receiver that he/she cancels the matching.

A matching reconstruction is initiated by somebody's sending a Divorce message after a matching game is completed. The rejected member will propose to another member to get a new matching. When the rejected member tries to get another partner using the man-optimal/woman-optimal GS-algorithm [7], he/she may obtain a new partner, and then the new matching is also stable. If the matching reconstruction is completed successfully, the initiator of the matching reconstruction gets a preferable partner.

3. Stabilization for Varied PL

In the previous works, we assumed the preference lists are fixed. However it is more natural that a member changes his/her PL during matching game because of his/her autonomous behavior. In this section, we allow preference lists to be changed at some stable matching. We note here that when a PL is varied at a stable matching M, the stability may be broken.

Now we consider a situation in which matching M becomes unstable. We suppose a member p changes his/her PL at matching M. In this case, if a member q who prefers p to q's partner in M is placed, by the changing, at a higher position in p's PL than p's current partner, the stability of M is broken. Note that q's position before changing should be lower than p's partner under the stability of M.

To obtain stable matching for the varied PL, the GS-algorithm can be employed again. However, it requires all the members to execute the algorithm and generally leads to quite different matching from M, even though the PL is changed in local. Moreover, the computation costs can be expensive.

Therefore, we propose an algorithm to stabilize a matching in which as few as possible members join to the matching stabilization, that is, the execution is performed locally (Figure 3). In Figure 3, function receiver is same in Figure 2.

In M, the relation between p and q causes unstability. q cannot recognize the fact that p's preference list has been changed, but p knows this of course. Then p sends Propose message to q and q sends Accept message to p while p and q also send Reject message to their current respect partners. The rejected members send Propose messages for getting new partner from the top of his/her PL. If the rejected members send Propose messages to each other, they become new pair and a new stable matching M' is obtained. Otherwise, if both of the rejected members get new partners, the partners at M of the new partners become new rejected members. If one of the re-

```
1  function proposer(var i: integer): integer
2  var p: integer;
3  begin i := i + 1;
4    repeat
5      p := the i-th member of PL;
6      Send(p, Propose);
7      Receive(p, Mes);
8      case Mes of
9        'Reject': i := i + 1;
10       'Accept': break the loop;
11     end;
12   until;
13   proposer := p;
14 end.
15
16 function receiver(var m: integer): integer
17 var i, p: integer;
18 begin
19   repeat
20     Receive(p, Mes);
21     case Mes of
22       'Propose':
23         if prefer p to m then
24           Send(m, Reject);
25           m := p;
26         else
27           Send(p, Reject);
28         end;
29       'Reject':
30         i := rank of m;
31         m := proposer(i);
32       'Proposed':
33         break the loop;
34     end;
35   until;
36   receiver := m;
37 end.
38
39 procedure Initiator;
40 var partner, m, p: integer;
41 begin m := current partner;
42   Send(m, Divorce);
43   repeat
44     Receive(p, Mes);
45     if prefer p to m then
46       Send(p, Accept);
47       partner := p;
48     else
49       Send(p, Reject);
50   until partner ≠ m;
51   Send(all non-initiator, Proposed);
52 end.
53
54 procedure Non-Initiator;
55 var m, partner: integer;
56 begin m := current partner;
57   partner := receiver(m);
58 end.
```

Figure 2. Matching Reconstruction

```
1  function proposer(var i: integer): integer;
2  var j, p: integer;
3  begin i := i + 1, j := the lowest rank + 1;
4    repeat
5      if i < j then
6        p := the i-th member of PL;
7        Send(p, Propose);
8      else
9        if j < the lowest rank + 1 then
10         p := i-th member;
11         Send(p, Accept);
12         Send(all members, Proposed);
13       end;
14     end;
15     Receive(p, Mes);
16     case Mes of
17       'Propose':
18         if rank of p ≤ i then
19           Send(p, Accept);
20           Send(all members, Proposed);
21         else
22           j := rank of p;
23         end;
24       'Reject': i := i + 1;
25       'Accept': break the loop;
26     end;
27   until;
28   proposer := p;
29 end.
30
31 procedure Changer;
32 var partner, i, j, m, p: integer;
33 begin m := current partner;
34   i := 1; j := rank of m;
35   repeat
36     if i < j then
37       Send(m, Divorce);
38     else break the loop;
39     end;
40     Receive(p, Mes);
41     case Mes of
42       'Reject': i := i + 1;
43       'Accept': Send(m, Reject);
44         m := receiver(p);
45         break the loop;
46   until
47   partner := m;
48 end.
49
50 procedure Non-Changer;
51 var m, partner: integer;
52 begin m := current partner;
53   partner := receiver(m);
54 end.
```

Figure 3. Stabilizing Algorithm

jected members gets new partner and the other does not get, one of the current members of the rejected members receives Reject message and then become new rejected member, and the other rejected member waits Propose message. This process is continued till all the members meet a partners. Consequently, we can lead to the next theorem.

Theorem 2 *The stabilizing algorithm in Figure 3 obtains a stable matching and it requires a minimal set of members to join the execution.*

4. Concluding Remarks

In this paper, we extended distributed stable marriage problem so that a preference list can be varied. We show the situation when a stable matching becomes unstable because of varying a PL. And then, we presented an algorithm to stabilize unstable matching by execution with a minimal set of members. As future works, we characterize formally the stable matching obtained by the proposed algorithm and evaluate the distance between the original matching and the obtained one.

References

[1] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," American Mathematical Monthly, Vol. 69, pp. 9-15, 1962.

[2] D. Gusfield and R. W. Irving, The Stable Marriage Problem Structure and Algorithms, The MIT Press, 1989.

[3] Donald E. Kunth, Stable Marriage and Its Relation to Other Combinatorial Problems, CRM Proceedings & Lecture Notes, Vol. 10, American Mathematical Society, 1997.

[4] M. Nakamura, K. Onaga, and S. Kyan, "Sex-fair Stable Marriage Problem and Its GA Solution," IEICE Trans. Fundamentals, Vol. E78-A, No.6, pp. 664-670, 1995.

[5] H. Kinjo, M. Nakamura, and K. Onaga, "Distributed Stable Marriage of Autonomous Mobile Robots and Battery Charger Station," IEICE Trans. Fundamentals, Vol. E79-A, No. 11, pp. 1856-1859, 1996.

[6] H. Nagamochi, M. Yamashita, and T. Ibaraki, "Distributed Algorithms for Cooperative Controlling of Anonymous mobile Robots," IEICE Technical Report, COMP95-24, pp. 31-40, 1995.

[7] H. Kinjo, M. Nakamura, and K. Onaga, "Autonomous Mechanism for Partner Exchanging in Distributed Stable Marriage Problem," IEICE Trans. Fundamentals, Vol. E80-A, No. 6, pp. 1040-1048, 1997.