An Improved Parallel Iterative Algorithm for Stable Matching

Colin White (Undergraduate)
colin.white2@gmail.com, Amherst College

Enyue Lu (Faculty Member)
ealu@salisbury.edu, Salisbury University

Abstract

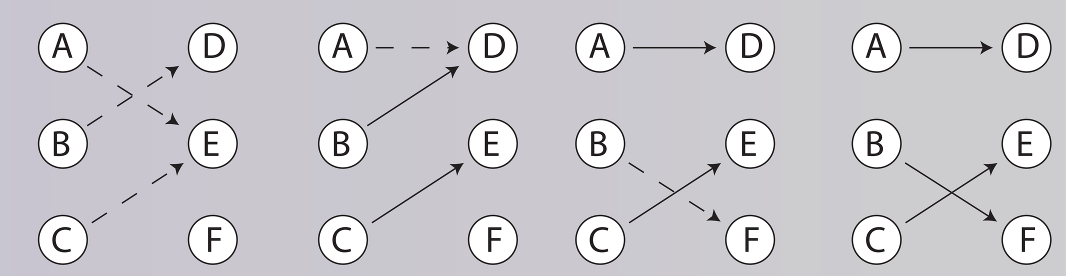
We propose a parallel algorithm for finding a stable matching that converges in *O(n* log *n)* average time using n^2 processors. The algorithm is based on the Parallel Iterative Improvement (PII) algorithm, which finds a stable matching with approximately a 90% success rate. Our algorithm, called the PII-SC algorithm, uses a smart initiation method that decreases the number of iterations to find a stable matching, and also applies a cycle detection method to find a stable matching based on patterns in the preference lists. Both methods decrease the number of times it fails to find a stable matching by three orders of magnitude, and when combined, the chance of failure is less than 10^{-7}.

Stable Matching

The stable matching (or stable marriage) problem was first introduced by Gale and Shapley. Given *n* men, *n* women, and *2n* preference lists in which each person ranks all members of the opposite sex by preference, a *matching* is an unordered set of *n* pairs of a man and woman in which each person is in exactly one pair. A matching is *unstable* if there exists a man and a woman who are not paired with each other, but both prefer each other to their current partner. Otherwise, a matching is *stable*. The stable matching problem has a wide variety of applications, from assigning doctors to hospitals, to real-time switch scheduling, to problems in economics.

The Gale-Shapley Algorithm

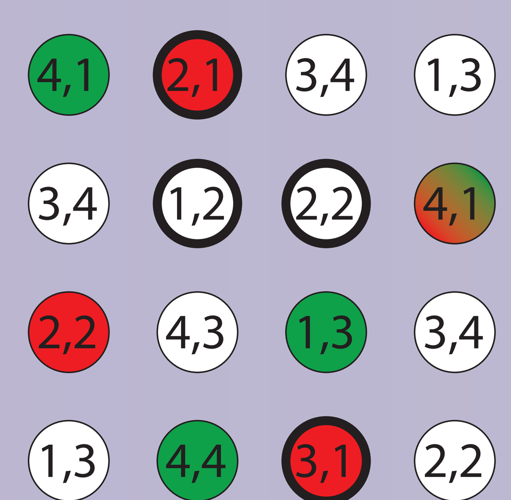
Preference Lists:
A: {E,D,F}
B: {D,F,E}
C: {E,F,D}
D: {A,B,C}
E: {B,C,A}
F: {C,A,B}



Gale and Shapley presented an $O(n^2)$ algorithm to find a stable matching from any preference lists, proving that a stable matching must always exist. Each man proposes to his highest-ranked woman who hasn't rejected him (dotted arrows); each woman chooses her highest-ranked proposal (solid arrows).

The PII Algorithm

INITIATION PHASE:
•Generate a random initial matching
ITERATION PHASE:
•Find unstable pairs. If there are none, return
•Pick highest-ranked unstable pair first by man's preference, then woman's preference to have one per row/column
•Fill out a matching with additional pairs
•Repeat iteration phase

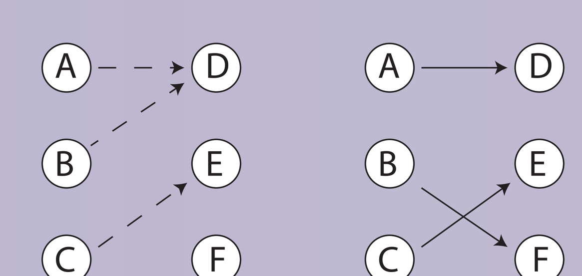


In each circle (x,y) in row i and column j, left value x is the preference of woman j for man i, and right value y is the preference rank of man i for woman j. Green pairs are in the current matching, unstable pairs are outlined in bold, and red pairs are in the new matching. The PII algorithm uses n^2 processors for an average time of O(*n* log *n*), and fails to find a stable matching roughly 10% of the time due to cycling in the iteration phase.

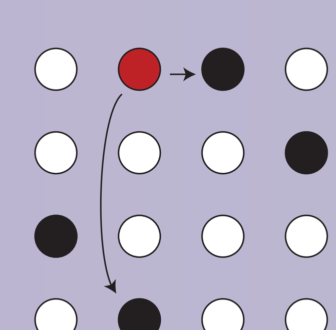
The PII-SC Algorithm

INITIATION PHASE:
•Run Gale-Shapley algorithm, however men are not allowed to propose to women that are currently in a pair
ITERATION PHASE:
•Same as PII for first *3n* steps
CYCLE DETECTION:
•Save $3n^{th}$ matching, check each subsequent iteration for cycling
•Save each unstable pair chosen by the algorithm, create chains of unstable pairs that share the same man or woman with the last pair in the chain
•Once the $3n^{th}$ matching is detected, pick every other pair from the chains and return

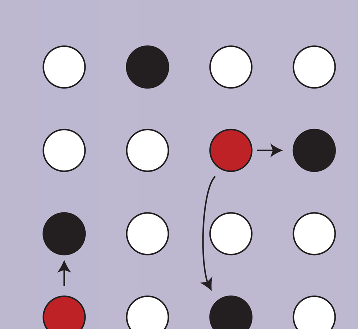
Preference Lists:
A: {D,F,E}
B: {D,E,F}
C: {E,D,F}
D: {A,B,C}
E: {A,B,C}
F: {B,C,A}



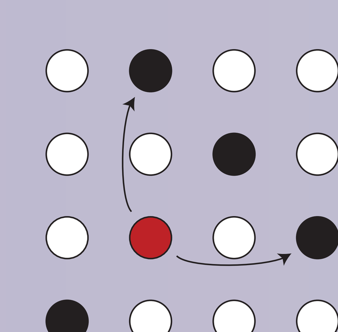
Smart initiation can be done in *O(n* log *n)* time with n^2 processors.



Iteration 1

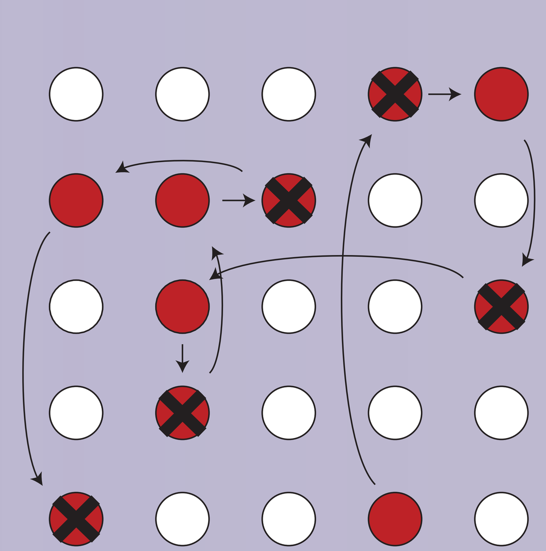


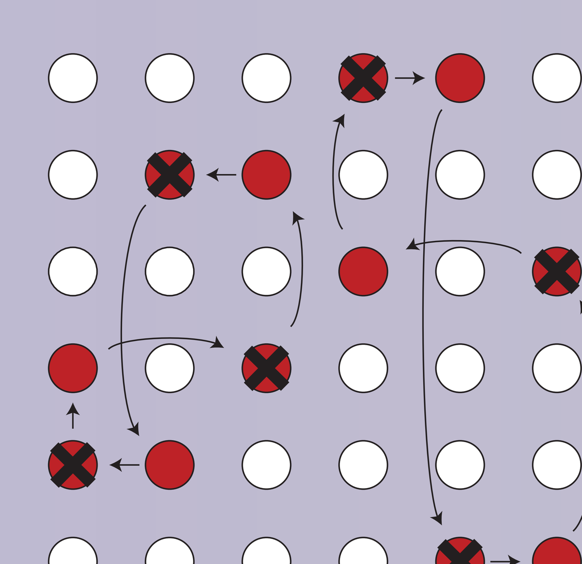
Iteration 2



Iteration 3

In the diagrams on the right, the rows represent men, and the columns represent women. The red circles represent unstable pairs chosen in the iteration phase, and the black circles represent current matchings. The arrows represent a pair replacing another pair in the matching during a step in the iteration phase. The stable matching consists of all pairs with x's.





Two examples of chains of unstable pairs, with their stable matchings

Results

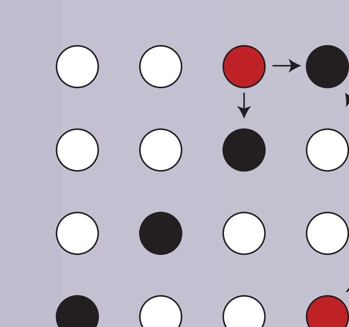
For each trial, we generate random preference lists for *n* men and women and run each of the different algorithms, outputting the success rate and the number of iterations taken to reach a stable matching. We ran the PII algorithm with the improvements (called the PII-SC algorithm) *10* million trials in order to calculate an accurate probability. For the rest of the algorithms, 100,000 trials was sufficient to calculate an accurate probability.

Table 1: Probability of finding a stable matching in 5n iterations with various n values

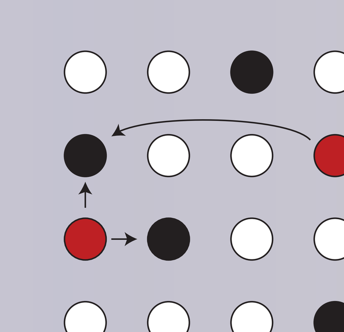
N	Gale-Shapley	Original PII	Smart Initiation	Cycle Detection	PII-SC Algorithm
10	0.9609	0.9809	0.9999	1.0000	1.0000000
20	0.8953	0.9513	0.9993	1.0000	1.0000000
30	0.8566	0.9226	0.9996	1.0000	0.9999998
40	0.8338	0.9007	0.9988	0.9999	0.9999996
50	0.8198	0.8801	0.9981	0.9999	0.9999994
60	0.8086	0.8719	0.9979	0.9998	0.9999994
70	0.8007	0.8650	0.9986	0.9998	0.9999992
80	0.7935	0.8553	0.9984	0.9998	0.9999989
90	0.7903	0.8642	0.9980	0.9997	0.9999992
100	0.7821	0.8579	0.9980	0.9998	0.9999985

Table 2: Number of successes for finding a stable matching with n=100 for various iterations per 100,000 trials

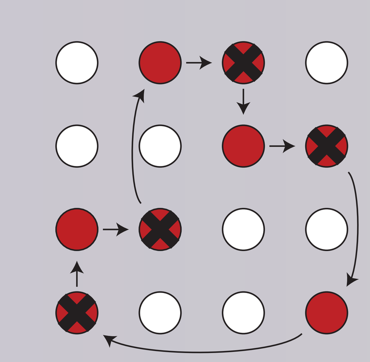
Iterations	Gale-Shapley	Original PII	Smart Initiation	Cycle Detection	PII-SC Algorithm
0.5n	93	81846	86166	81846	86166
n	5308	86363	95269	86363	95269
1.5n	19392	86425	99347	86425	99347
2n	35522	86427	99777	86427	99777
2.5n	50128	86427	99784	86427	99784
3n	61998	86427	99784	86427	99784
3.5n	71012	86427	99784	99981	100000
4n	78063	86427	99784	99982	100000
4.5n	83483	86427	99784	99982	100000
5n	87609	86427	99784	99982	100000



Iteration 4



Iteration 5



Stable Matching

The final diagram shows all of the unstable pairs throughout the four iterations. Picking every other pair in the chain results in a stable matching.

Ongoing Work

We are working on classifying why the cycle detection method sometimes fails to find a stable matching. If we can generalize the algorithm to work for all cases, prove that after *3n* iterations the algorithm must be cycling, and prove that every cycle length is at most *2n*, then we will have an *O(n* log *n)* algorithm for stable matching with n^2 processors. We are also working on finding a linear time algorithm with n^3 processors.

Acknowledgements

This work is funded by NSF CCF-1156509 under Research Experiences for Undergraduates Program. C. White did his work as an REU (Research Experiences for Undergraduates) student at Salisbury University during the summer of 2013.