An Improved Parallel Iterative Algorithm for Stable Matching

Colin White, Undergraduate
Depts. of Math & Computer Science
Amherst College
Amherst, USA
colin.white2@gmail.com

Dr. Enyue Lu, Faculty Mentor
Dept. of Math. & Computer Science
Salisbury University
Salisbury, USA
ealu@salisbury.edu

***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}.**

***Keywords*—Stable Matching, Stable Marriage, Graph Algorithms, Parallel Algorithms**

I. INTRODUCTION

The stable matching (or stable marriage) problem was first introduced by Gale and Shapley [1]. 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*. Gale and Shapley presented an $O(n^2)$ algorithm to find a stable matching from any preference lists, thus proving that a stable matching must always exist. The stable matching problem has a wide variety of applications, from assigning doctors to hospitals, to real-time switch scheduling, to problems in economics [2,6].

II. PII ALGORITHM

The original PII algorithm was designed for time-sensitive switch scheduling network problems [4]. It consists of two phases: the initiation phase and the iteration phase. It starts by generating a random initial matching in the initiation phase, and then iteratively picks the "most" unstable pairs, collectively called the set NM_1 of *type-1 new matching pairs* (*nm_1-pairs*), to replace current matching pairs in the iteration phase. Since each nm_1-pair contains a man and a woman from two different current matching pairs, often other pairs must be added until we have *n* pairs again. Such pairs, which make up the set NM_2 of *type-2 new matching pairs* (*nm_2-pairs*), are sometimes difficult to find because of long chains of nm_1-pairs that overlap with respect to the replaced pairs. The nm_2-pairs, which may not be favorable, can cause new unstable pairs in the new matching, which in turn allows the possibility of cycling in the iteration phase. The PII algorithm has a 90% chance of finding a stable matching within *n* iterations (the vast majority of failures are due to cycling), and runs in $O(n \log n)$ average time with n^2 processors [3,4].

III. NEW METHODS

In this paper, we give two major improvements that increase the performance of the PII algorithm: smart initiation and cycle detection.

A. Smart Initiation

The first major improvement is in the initiation phase: we use a simple method to find an initial matching that performs much better than a random matching. We accomplish this in a way similar to the Gale-Shapley algorithm: pair each man with their top-ranked woman, breaking ties with the woman's preference. Then each unpaired man gets paired with their next best choice, however, unlike in the Gale-Shapley algorithm, they cannot propose to women that are currently already paired with someone. We continue in this way until we have a matching.

We claim that the smart initiation method can be done in linear time with n^2 processors as follows. There are at most *n* steps before every person is provably paired up. During each step, there are at most *n* proposals from the men. Every woman can pick her most preferable proposal in constant time using constant-time find-minimum algorithm, which finds the minimum of *k* elements using k^2 processors [5]. This causes every step to take constant time, making the whole method run in linear time.

The smart initiation method decreases the chance of the algorithm failing to find a stable matching from *1* in *10* to *1* in *20*. Furthermore, when the method is run in parallel with the woman-optimal version of the same method (in which the roles of men and women are switched), the chances of failing are reduced to roughly *1* in *400*.

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

B. Cycle Detection

The second major improvement is in the iteration phase: an additional step that detects if the algorithm is cycling and then outputs the stable matching. When the algorithm has reached *3n* iterations without finding a stable matching, experimentation has shown that the matchings generated by the iteration phase must be cycling. It then saves the current matching and checks the matching generated in each successive iteration to see if it is the same as the one in the $3n^{th}$ iteration. Meanwhile, the algorithm starts creating chains of nm_1-pairs: each nm_1-pair either starts a new chain, or is added to an existing chain if it shares the same man or woman with the last pair that was added. After the $3n^{th}$ matching is detected again, we check if any nm_1 chains can merge together into nm_1-*cycles*. We then pick every other nm_1 pair in the cycles, which results in a stable matching roughly *9,999* times out of *10,000*. The cycle detection phase can be done in constant time, which does not change the runtime of the full algorithm.

IV. Results and Discussion

We have implemented the PII algorithm with the two major improvements (which we call the PII-SC algorithm), along with the PII algorithm with smart initiation, the PII algorithm with cycle detection, the original PII algorithm, and the Gale-Shapley algorithm for comparison. For each trial, we generate random preference lists and run each of the different algorithms, outputting the success rate and the number of iterations taken to reach a stable matching. In table 1, we calculate the success rate for *5n* iterations, varying *n* from *10* to *100*. We ran the PII-SC algorithm for *10* million trials in order to calculate an accurate probability, but to save computing time, we deemed *100,000* trials each was sufficient for the rest of the algorithms. Each of the two major improvements decreases the probability of failure by multiple orders of magnitude. In table 2, we calculate the number of successes with *n=100*, varying the number of iterations from *0.5n* to *5n*. The cycle detection algorithm and PII-SC algorithm only start looking for cycles after *3n* iterations, so we see a major increase in the rate of success at that time.

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

V. Ongoing Work

We are working on classifying cases where the cycle detection algorithm sometimes fails to find a stable matching. We have identified different categories of "bad" cases and why each of them fails to output a stable matching. If we can generalize the cycle detection algorithm to account for these cases, along with proving that an initial matching must either become stable or cycle after *3n* iterations, and that the longest cycle length is *2n*, then we would have an $O(n \log n)$ algorithm for stable matching with n^2 processors. We are also working towards finding a similar algorithm that uses n^3 processors and runs in linear time.

References

[1] Gale, D. and Shapley, L.S.: "College admissions and the stability of marriage", *American Mathematical Monthly*, Vol. 69 (1962) 9–15

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

[3] Korakakis, E.: *Examining the Parallelization Limits of the Stable Matching Problem*, Master's Thesis, University of Edinburgh (2005)

[4] Lu E. and Zheng S. Q.: "A Parallel Iterative Improvement Stable Matching Algorithm", *High performance computing, Lecture Notes in Computer Science*, Springer-Verlag, Vol. 2913 (2003) 55-65

[5] Quinn, Michael J.: *Parallel Computing: Theory and Practice*, 2nd ed., New York: McGraw-Hill, 1994

[6] The Prize in Economic Sciences 2012, "Stable matching: Theory, evidence, and practical design", http://www.nobelprize.org /nobel_prizes/economic-sciences/laureates/2012/popular-economicsciences2012.pdf