Improved Parallel Stable Matching

Caleb Clark (caleb.clark@my.wheaton.edu)*,Enyue Lu (ealu@salisbury.edu)**

*Wheaton College,**Salisbury University

Abstract

We present convergent parallel iterative improvement (CPII), a stable matching algorithm that runs on n^2 processors. CPII improves on parallel iterative improvement (PII), another stable matching algorithm on n^2 processors by converging to a stable matching in cases where PII did not converge, and converging more quickly than PII. We also have simulations that suggests CPII may outperform current stable matching algorithms that use n^2 processors or less.

Stable Matching Problem

Consider a group of n men and a group of n women in which each woman has strictly ordered every man based on her preference for him and each man has also strictly ordered every woman based on his preference for her. The stable matching problem asks: how can we match these two groups so that no two people prefer each other to their current partner? That is to say, how can we match each man to a woman such that if any woman prefers a man to her partner, then he prefers his partner to her. The stable matching problem was first proposed by Gale and Shapely who also introduced an algorithm (GS) which always finds a stable matching in, at worst, $O(n^2)$ time. It was later shown that GS takes $O(n \lg n)$ time on average.

The Gale-Shapley Algorithm (GS)

Everyone starts single. A single man proposes to his favorite woman, and partners with her. Then another single man proposes to his favorite woman who has not already rejected him. If she is single she partners with him, otherwise she compares him to her current partner, partners with her favorite of the two, and rejects the other. This process is repeated until no one is single. The end result is always a stable matching.

Parallel Approaches and NPGS

There has been a substantial amount of interest in parallel algorithms to solve the stable matching problem for many years, but very little progress.The GS algorithm has an obvious parallelization which we will call Naive Parallel GS (NPGS). NPGS is identical to GS, but in each iteration rather than just having one man propose, every single man proposes simultaneously. NPGS runs on n processors, and takes $O(\lg n)$ per iteration since each woman needs to search for her favorite man out of all the men that proposed to her.

PII and PII-SC algorithms

Parallel iterative improvement (PII) is a computational procedure that runs on n^2 processors and often finds a stable matching quickly, however, about 13% of the time, PII cycles and does not converge to a stable matching . PII-SC was an attempt to fix PII to prevent it from cycling that succeeded in increasing the probability of finding a stable matching, however PII-SC still cycles in some rare cases (<1%). PII-SC also adds a $O(n)$ initialization step that it must run before it begins iterating.

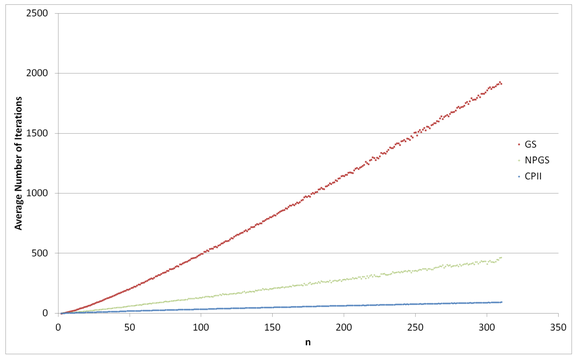

Figure 1: Graphs of the average number of iterations over 1000 trials for various values of n

Convergent Parallel Iterative Improvement (CPII)

CPII is an algorithm that combines the fast parallel techniques of PII and PII-SC with the fast algorithmic techniques of GS. We are able to prove that CPII converges in $O(n^2)$ iterations. In addition, our simulations show CPII usually converges as fast as the PII and PII-SC algorithms. It works as follows.

Step 0: Start with all individuals being single.

Step 1: Find A, the set of all pairs who prefer being with each other to being with their current partners. If A is empty, then terminate (Note: everyone prefers having a partner to being single).

Step 2: Find $B \subseteq A$, a set of pairs in A with no repeated males. If there are multiple pairs in A that contain the same male we eliminate all the pairs except the one that that male prefers the most.

Step 3: Find $C \subseteq B$, a set of pairs in B with no repeated females. If there are multiple pairs in B that contain the same female we eliminate all the pairs except the one that that female prefers the most.

Step 4: Match all the pairs in C with each other. If this breaks up any other couples leave them single for the start of the next round. Go to step 1.

Comparison with GS and NPGS

We compare CPII, NPGS, and GS graphically in figures 1 and 2 based on data from our simulations. Both NPGS and CPII iterations take $O(\lg n)$ time, while a GS iteration takes constant time.

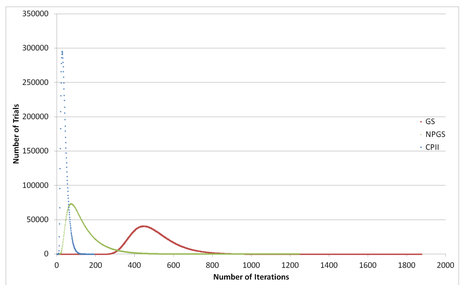

Figure 2: Graphs of the number of iterations each of 10 million trials took with n=100

Convergence of CPII

When examined carefully, CPII can rephrased in terms of GS as follows. For every iteration of CPII, every single man proposes to his favorite woman who *will not* not reject him. This relationship allows us to prove that CPII will always find a stable matching. This improves on GS, where each single man proposes to his favorite woman who *has not already* rejected him.

Comparison with PII and PII-SC

We compare PII and PII-SC to CPII based on how quickly they converge. It should be noted that this table does not account for initialization time, which is important since CPII, PII, and PII-SC take $O(1)$,$O(\lg n)$, and $O(n)$ time respectively to initialize before they begin iterating. For all 3 programs an iteration takes $O(\lg n)$ time.

iterations	PII	PII-SC	CPII
0.5n	81846	86166	81459
n	86363	95269	99652
1.5n	86425	99347	100000
2n	86427	99777	100000
3n	86427	99784	100000
4n	86427	100000	100000

Table 1: Number of successes for finding a stable matching with n=100 for various iterations per 100,000 trials

Conclusion

In conclusion, CPII improves on the PII algorithm, and reveals new information about the parallelizability of the stable matching problem. In the future, we hope to find and prove the average runtime of CPII. Additionally, we plan to explore how CPII performs on difficult (NP-hard) matching problems such as finding the maximum size matching when people have incomplete preferences with ties.

Acknowledgements

This work is funded by the National Science Foundation under the Research Experience for Undergraduates Program CNS-1757017 grant.