Parallel Iterative Improvement Stable Matching Algorithms

Blake Hurlburt[1], Ryan Yost[2], Dr. Enyue Lu[3]
[1] University at Buffalo [2] Hampden-Sydney College [3] Salisbury University

Abstract

The stable matching problem has many practical applications to real-time systems. However, the existing algorithms that solve this problem are infeasible for these systems because they require too much time. Previous work on algorithms with an acceptable time bound are unable to compute a stable matching for all inputs.

We propose a parallel iterative improvement algorithm that solves the stable matching problem with a higher success rate than previous algorithms within $O(n\ log\ n)$ runtime using n^2 processors. We address some unsolved cases enumerated by previous work on this problem. However, our algorithm is unable to determine a stable matching with 100% success rate; further investigation of a new method is required.

Introduction

The stable matching problem was first introduced by Gale and Shapley in 1962. A *matching* a set of pairs of men and women in which each man and woman is part of exactly one pair. An *unstable pair* is a man and woman who are not paired together, but both prefer each other to their current partners. A matching is *stable* if the matching has no unstable pairs.

One real-time application of solutions to the stable matching problem is switch scheduling for packet/cell switches. While the Gale-Shapley algorithm serves as a proof that a stable matching exists for any group of people, its $\Theta(n^2)$ runtime makes it infeasible for use in real-time systems.

Prior work in developing a parallel iterative improvement algorithm to solve this problem has been done by Dr. Enyue Lu and Colin White. The algorithm has two phases: *initiation* and *iteration*. In the initiation phase, an initial matching is derived for the men and women. In the iteration phase, a better matching is derived from the existing matching. The iteration phase continues until a matching is successfully found.

Our goal is to find an implementable algorithm that can determine a stable matching within $O(n\ log\ n)$ runtime using n^2 processors. This runtime and required amount of computational resources is acceptable for real-time systems.

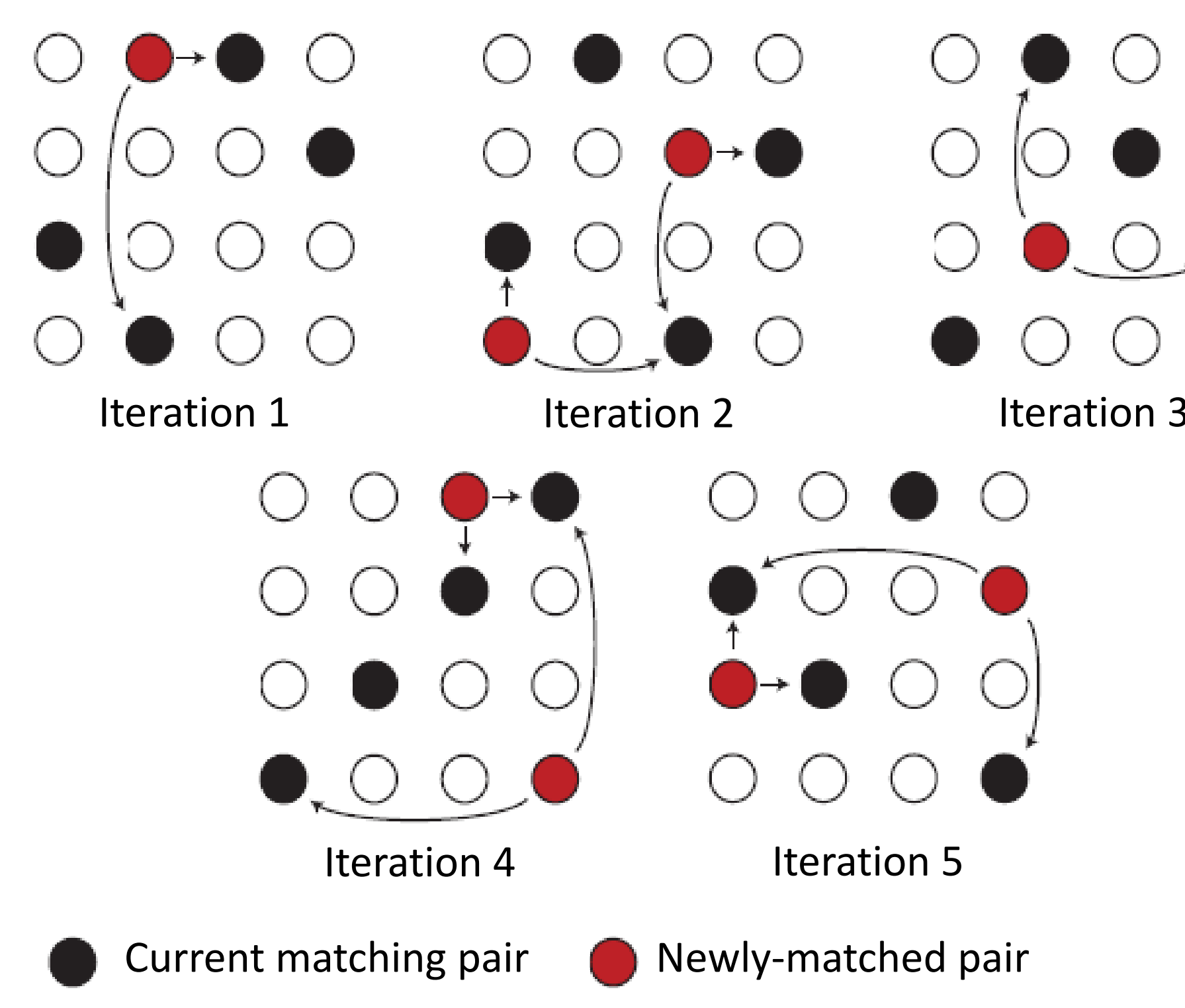

Figure 1. An example of the iteration phase. Each matching is derived from the previous matching until the matching is stable.

Problem

The original algorithm failed at a rate of approximately 10% due to *cycling*. This occurs when the matching derived in the iteration phase is a matching that has already been visited. Further work introduced *cycle detection* to mitigate this problem. For the majority of cycles, simply selecting every other pair in the cycle to be part of the derived matching will yield a stable matching. While these improvements reduce the rate of failure to approximately 1/400, this does not work for irregularly formed cycles.

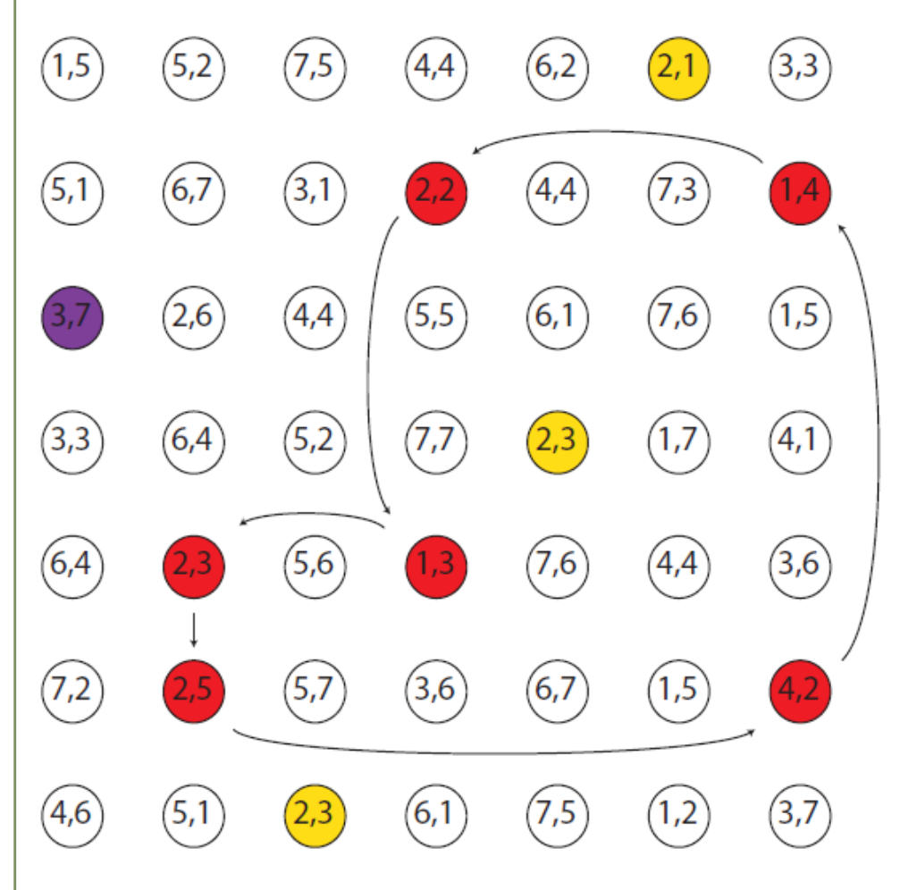

Figure 2. An example of a hole. In this case, a stable matching can be derived by filling in the hole.

The major irregularity occurs when the iteration step results in a *hole*, where a man and a woman are left unpaired. Other cases of irregularity in the formation of cycles include *two cycles*, *figure eight*, *three in a row*, *dead ends*, and *long cycles*. Two cycles simply means that two independent cycles are formed. A figure eight is when two cycles form with one pair in common. Three in a row occurs when one person is part of the cycle three times. A dead end is when the cycle splits into two possible paths, where one is a dead end. A long cycle is a cycle of length at least $2n$.

Objective

The purpose of our work is to find an algorithm that can correct the irregular cases left by previous work in order to achieve 100% success in finding a stable matching. In order to do this, we will need to be able to detect each type of irregularity and be able to derive a matching from these cases. This algorithm will have a runtime of $O(n\ log\ n)$ and use n^2 processors.

Results

Our algorithm is an improved version of the prior algorithms that accounts for cases in the cycle detection that were not previously handled. After having classified these types of cycles which could not be solved by previous algorithms through Colin White's work and our own battery of tests, we developed improvements to the previous algorithm that would determine when one of these cases would occur and apply the correct solution.

We were able to resolve cases in which two cycles, figure eights, three in a row, and dead ends occur. For two cycles, all four combinations of selecting every other pair result in a stable matching. Figure eights are corrected by simply ensuring no pair is selected twice. When three pairs in a row are part of the cycle, selecting the center pair ensures that no two pairs from the same row are selected, resulting in a stable matching. Finally, when a dead end occurs, we select the first pair in the row of three in order to derive a valid matching.

Conclusions

We were unable to resolve cases in which long cycles and certain types of holes appear. For long cycles, certain pairs must be skipped when selecting members of the new matching, but it is unclear which pairs should be skipped and which should be selected. In most occurrences, holes can be corrected by simply pairing the two unpaired people, resulting immediately in a stable matching. However, filling in the hole occasionally results in another cycle. At this point, the newly created cycle must be resolved; however, this may lead to a state that has already been visited.

Because we were unable to resolve these cases, our improved algorithm was unable to derive a stable matching with 100% success rate. Although the improvements reduce the probability of failure, further analysis will be required to resolve all irregularity cases and achieve a 100% success rate.

Acknowledgements

This work is funded by NSF CCF-1460900 under Research Experiences for Undergraduates Program. B. Hurlburt and R. Yost did their work as REU students at Salisbury University during the summer of 2017.