



Classifier Architectures for Stable Matching Algorithms

Matthew Goldman, Binghamton University; Juniper Pasternak, Kalamazoo College; Mentor: Dr. Enyue Lu, Salisbury University



Background

Given n men and n women: a matching is defined as an assignment from each man to exactly one woman. A matching is stable if there do not exist a man and woman who prefer each other over their current partners.

Each person ranks members of the opposite sex in order of preference. A view of the rankings in the stable matching problem is through a ranking matrix which is an $n \times n$ matrix containing tuples, where the left value at row i , column j , reflects man i 's ranking of woman j and the right value is woman j 's ranking of man i .

The stable matching problem has applications in medical residency programs and switch scheduling; and has been researched in the fields of economics, mathematics, and computer science.

The Parallel Iterative Improvement algorithm (PII) is an efficient algorithm to solve the stable matching problem that uses n^2 processors, but it fails to converge (cycles) about 14% of the time for large n .

CPII is a simpler variant of the PII algorithm that always converges. CPII contains an arbitrary decision before the algorithm begins for how to select pairs to build towards a stable matching, resulting in two variants of the algorithm.

Problem

PII can cycle, so it may be beneficial to train a machine learning model to be a cycle predictor since it will be able to tell the PII algorithm to start with a different random matching and save time that PII would have spent stuck in a cycle. Thus, our goal is to apply machine learning models to the problem of finding a stable matching by creating cycle prediction classifiers, which could improve the convergence rate of PII.

CPII contains two variants of the algorithm. These two variants are not guaranteed to find a stable matching in the same number of iterations. We can create a machine learning model that predicts which variant takes less iterations to optimize the algorithm. This model can save iterations by telling the algorithm to perform the second variant of the algorithm when it will find a stable matching in less iterations.

PII Cycle Prediction Model

Each pair in a PII iteration is directly related to all pairs in the same row and column. Because we have a combination of sequential input and data that contains contextual relationships, we decided to try using a transformer model. This allows us to take advantage of self-attention to find contextual relationships between the data. Because we are attempting a classification problem, we used an encoding-only transformer.

CPII Variant Selection Model

We also chose to use a transformer with CPII variant selection. Using a transformer with full self-attention takes $O(n^2)$ time, which is worse than CPII's average case runtime on the same number of processors. Instead of using full self-attention, we developed attention methods that are $O(n)$ on n^2 processors. These are row-col attention, pair types attention, and combined attention, which are described in Fig. 1. We use row-col attention because it is the most efficient.

Because a variant of the algorithm can save different numbers of iterations each time it is run, we implemented a weighted loss function. We weighted the loss based on the difference in iterations between the two variants. We decided to use a logarithmic function to scale our weights when training, due to its high performance.

As shown in Table 1, we developed iteration accuracy and speedup metrics to evaluate the performance of the model. Iteration accuracy is the percentage of all possible iterations the model saves by predicting which variant is faster. Aggregate speedup measures how much faster the algorithm would take when using the predictions compared to running normally on the entire dataset. Average speedup measures the average of how much faster each instance of running the algorithm is. The speedup ratios compare these speedup values to the maximum possible speedups that would occur if the model achieved 100% accuracy.

CPII Self-Attention

Ranking matrix				Row-Col Attention			
(1, 1)	(4, 4)	(3, 2)	(2, 2)	(1, 1)	(4, 4)	(3, 2)	(2, 2)
(2, 4)	(3, 3)	(4, 4)	(1, 3)	(2, 4)	(3, 3)	(4, 4)	(1, 3)
(4, 3)	(1, 2)	(3, 3)	(2, 4)	(4, 3)	(1, 2)	(3, 3)	(2, 4)
(3, 2)	(2, 1)	(1, 1)	(4, 1)	(3, 2)	(2, 1)	(1, 1)	(4, 1)
Pair-Types Attention				Combined Attention			
(1, 1)	(4, 4)	(3, 2)	(2, 2)	(1, 1)	(4, 4)	(3, 2)	(2, 2)
(2, 4)	(3, 3)	(4, 4)	(1, 3)	(2, 4)	(3, 3)	(4, 4)	(1, 3)
(4, 3)	(1, 2)	(3, 3)	(2, 4)	(4, 3)	(1, 2)	(3, 3)	(2, 4)
(3, 2)	(2, 1)	(1, 1)	(4, 1)	(3, 2)	(2, 1)	(1, 1)	(4, 1)

Figure 1. Types of self-attention given the ranking matrix with the entry performing attention **bolded**. Bolded values in attention matrices represent values that will be queried.

PII Cycle Prediction Results

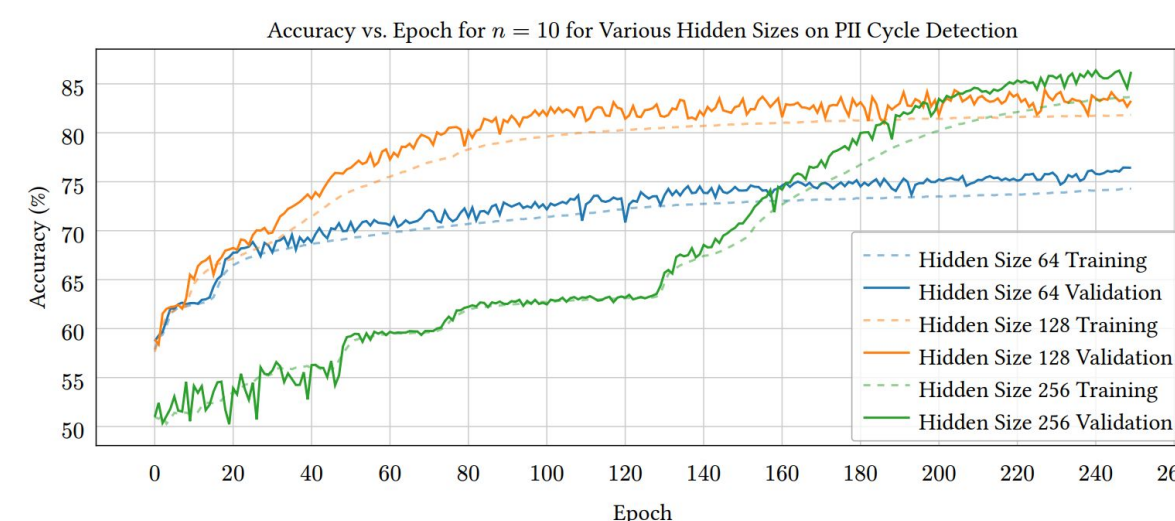


Figure 2. Evaluation of the transformer model for PII cycle prediction on $n=10$ data of length 2M for various hidden sizes.

CPII Variant Selection Results

n	Data Len.	Hidden Size	# of Layers	Weighted Loss Function	Acc. (%)	Iteration Acc. (%)	Aggregate Speedup	Average Speedup	Aggregate Speedup Ratio (%)	Average Speedup Ratio (%)
10	1M	128	4	None	88.6	87.2	1.25	1.32	84.0	92.2
10	1M	128	4	Log	89.1	88.0	1.25	1.32	85.0	91.8
10	2M	128	4	Log	89.4	88.5	1.25	1.32	85.6	92.2
20	2M	64	8	Log	88.1	89.4	1.26	1.33	86.6	92.9
20	4M	64	8	Log	89.8	91.4	1.27	1.34	89.1	94.5

Table 1. Comparison of our best models for CPII variant selection on $n=10$ and $n=20$ data of varying lengths. All models use row-col self-attention. Best results for each metric for each n are bolded.

Conclusions and Future Work

Using machine learning, we are able to accurately predict whether PII will cycle for small values of n , but still struggle to predict cycles for large values of n .

For the CPII variant selector, we found that using a transformer model with a logarithmic weighted loss function and row-col attention performed the best. We performed preliminary tests on $n=30$, and noticed that to obtain high performance, the size of the model must scale, potentially hindering practicality.

We want to eventually deploy the CPII variant selector in a parallel implementation of CPII. This would allow us to more accurately determine how much the model would improve the efficiency of CPII.

References

- [1] Caleb Clark and Enyue Lu. 2018. Improved Parallel Stable Matching. (2018).
- [2] Luke Ditria. 2023. Pytorch Tutorials. Retrieved from https://github.com/LukeDitria/pytorch_tutorials
- [3] D. Gale and L. S. Shapley. 1962. College Admissions and the Stability of Marriage. *The American Mathematical Monthly* 69, 1 (1962), 9–15. Retrieved June 30, 2026 from <http://www.jstor.org/stable/2312726>
- [4] Kazuo Iwama and Shuichi Miyazaki. 2008. A Survey of the Stable Marriage Problem and Its Variants. In *International Conference on Informatics Education and Research for Knowledge-Circulating Society (icks 2008)*, 2008. 131–136. <https://doi.org/10.1109/ICKS.2008.7>
- [5] Donald E. Knuth. 1997. *Stable Marriage and Its Relation to Other Combinatorial Problems*. American Mathematical Society.
- [6] Enyue Lu and S. Q. Zheng. 2003. A Parallel Iterative Improvement Stable Matching Algorithm. In *High Performance Computing - HPC 2003*, 2003. Springer Berlin Heidelberg, Berlin, Heidelberg, 55–65.
- [7] Poorya Mianjy, Raman Arora, and Rene Vidal. 2018. On the Implicit Bias of Dropout. Retrieved from <https://arxiv.org/abs/1806.09777>
- [8] Sai Srivatsa Ravindranath, Zhe Feng, Shira Li, Jonathan Ma, Scott Duke Kominers, and David C. Parkes. 2021. Deep Learning for Two-Sided Matching. *CoRR* abs/2107.03427, (2021). Retrieved from <https://arxiv.org/abs/2107.03427>
- [9] Zelin Wang and Jyh-Charn Liu. 2019. Translating Math Formula Images to LaTeX Sequences Using Deep Neural Networks with Sequence-level Training. Retrieved from <https://arxiv.org/abs/1908.11415>
- [10] Colin White and Enyue Lu. 2013. An Improved Parallel Iterative Algorithm for Stable Matching. 2013. SuperComputing.
- [11] Scott Wynn, Alec Kyritsis, Stephora Alberi, and Enyue Lu. 2024. Selection Improvements on the Parallel Iterative Improvement Algorithm for Stable Matching. *International Journal of Computer Applications* 186, 43 (September 2024), 49–56. <https://doi.org/10.5120/ijca2024924060>

Acknowledgements

Small parts of our code were based on repositories from [9], a publication, and [2] under MIT License.

This research was conducted through the EXERCISE REU project. The EXERCISE REU project is funded by the National Science Foundation grant number 2447041 with additional support from Salisbury University, the University of Maryland Eastern Shore, the Johns Hopkins University, the University of Maryland College Park, and the University of Maryland Baltimore County.