Clearing Kidney Exchanges via Graph Neural Network Guided Tree Search (Student Abstract)

Zeyu Zhao,[1,2] **John P. Dickerson**[2]
[1]Montgomery Blair High School, 51 University Blvd E, Silver Spring, MD 20901
[2]University of Maryland
zachzhao@umiacs.umd.edu, john@cs.umd.edu

Abstract

Kidney exchange is an organized barter market that allows patients with end-stage renal disease to trade willing donors—and thus kidneys—with other patient-donor pairs. The central clearing problem is to find an arrangement of swaps that maximizes the number of transplants. It is known to be NP-hard in almost all cases. Most existing approaches have modeled this problem as a mixed integer program (MIP), using classical branch-and-price-based tree search techniques to optimize. In this paper, we frame the clearing problem as a Maximum Weighted Independent Set (MWIS) problem, and use a Graph Neural Network guided Monte Carlo Tree Search to find a solution. Our initial results show that this approach outperforms baseline (non-optimal but scalable) algorithms. We believe that a learning-based optimization algorithm can improve upon existing approaches to the kidney exchange clearing problem.

1 Introduction

Transplantation is favored over dialysis as a treatment for chronic kidney disease. Although there is a supply of kidney transplants from cadavers, this resource is extremely limited. As of 2018, there are over 40,554 people in the U.S. entering the waiting list for a kidney cadaver transplants, with only 14,725 people leaving with a transplant.[1] If a patient has a willing donor, an operation may be performed if the two are compatible. Compatibility is influenced by a variety of factors, including blood type and antibodies within the patient's blood. However, in many cases, a patient has an incompatible donor, and cannot directly perform a transplantation.

Kidney exchange provides a method to match patients and donors. The pool of patients is commonly expressed with a compatibility graph G, with a vertex for each patient-donor pair. A directed edge e connects v_i to v_j if the patient of pair v_j is compatible with the donor of v_i. A solution can be achieved through cyclic trades, where each pair's donor donates to the next pair's patient. Our goal is to maximize the cardinality of pairs covered by the disjoint cycles.

This problem is known as the disjoint cycle covering problem, and is NP-hard for finite cycles of length at least three (Abraham, Blum, and Sandholm 2007). Both the theory and practice of kidney exchange have been impacted by the AI, OR, and Economics communities (Anderson et al. 2015; Dickerson and Sandholm 2015). We reformulate the disjoint cycle covering problem as Maximum Weighted Independent Set (MWIS). We create a new graph with a vertex for each possible cycle in the compatibility graph and weight set to its original cycle's cardinality. We create an edge between two vertices if their respective cycles share a vertex. Our goal is to select a subset of vertices in this new graph with (i) maximum weight and (ii) no two vertices sharing an edge.

Traditionally, these problems are solved through branch-and-price-based methods (Barnhart et al. 1998), relying on generic heuristics to guide the search. These algorithms often fail to exploit the distributional data (e.g., over the family of directed compatibility graphs) available via historical matchings. In other words, the constraints induced by the compatibility graph can be thought of as being drawn from some underlying distribution.

Motivated by this circumstance, we approach this problem (now cast as MWIS) as a learning problem. We adopt a graph neural network (GNN) as our heuristic, as it handles structured data well, and can scale to different sizes of input (i.e., graph sizes). Graph neural networks have seen success in a variety of areas, such as structured molecular prediction, text classification and relation reasoning (Wu et al. 2019). We then use the trained GNN as a heuristic in a Monte Carlo Tree Search (MCTS). MCTS-based approaches can work well in combinatorial games such as Go, and many combinatorial optimization problems (Browne and others 2012). We give promising preliminary results and future directions.

2 Methods

After formulating the compatiblity graph $G = (V, E)$ as a WMIS problem graph G, we adopt a Monte Carlo Tree Search (MCTS) guided by a Graph Neural Network (GNN).

Graph Neural Network

We use a graph neural network (GNN), $f_\theta(G)$, to predict the likelihood of a vertex being in an optimal solution. The input to the GNN is a weighted MWIS graph, $G = (V, E)$, with $n = |V|$. The graph neural network outputs a probabil-

[1]https://optn.transplant.hrsa.gov/data/

ity map $p \in [0,1]^n$, with each element p_i corresponding to vertex i's likelihood. We use supervised learning with an oracle to train the graph neural network. The oracle, for each graph G, outputs a binary label vector $\mathbf{1} \in \{0,1\}^n$ for a particular optimal solution, with 1 indicating that the corresponding vertex is in the solution, and 0 otherwise. We use the ADAM optimizer to minimize the cross entropy loss between the probability map $p = f_\theta(G)$ and $\mathbf{1}$.

Monte Carlo Tree Search

We adopt a Monte Carlo Tree Search (MCTS) guided by a GNN to tackle the MWIS problem. The GNN serves as a probabilistic prior for the search algorithm, narrowing down the large branching factor. We use the GNN in two distinct ways: to rank the unexplored nodes in the search tree for expansion, and as a coefficient in our upper confidence bound.

Selection. The MCTS algorithm follows a tree policy from the root node to a leaf node, a node not yet fully expanded. At step t of selection, the algorithm chooses $a_t = \text{argmax}_a(Q(s_t,a) + U(s_t,a))$, with Q as an estimation of the value of the state-action, and U as an upper confidence bound. We incorporate the GNN f_θ as a coefficient of U:

$$U(s_t,a) = c * P(s_t,a) * \frac{\sqrt{\sum_b N(s_t,b)}}{1+N(s_t,a)}$$

with $P(s,a) = [softmax(f_\theta(s))]_a$, and $N(s,a)$ as the state-action visit count. Using the GNN's coefficient allows the GNN to have an influence on the tree policy early on, with diminishing influence as the MCTS algorithm becomes more confident (indicated by the visit counts).

Expansion. Once the MCTS algorithm reaches a leaf node s_l, the algorithm picks a node to expand by choosing action

$$a_l \sim softmax([[f_\theta(s_l)]_a | a \in A_{unexplored}(s_l)])$$

This technique effectively allows the GNN to "rank" the unexplored nodes. Once a node has k explored nodes, we label the node as fully expanded. We use the GNN and this limit to manage the large branching factor that arises from the MWIS MDP.

Simulation. We use uniformly random rollouts starting from the leaf node to a terminal node s_l. Random rollouts provide a quicker simulation than a full rollout guided by the GNN, while retaining a reasonable amount of performance.

Backpropagation. Most MCTS algorithms use an average of rollouts to estimate a node's Q value. We propose to use a *maximum* of the rollouts, as our rollouts can be viewed as a lower bound on the node's true Q value (Sabharwal, Samulowitz, and Reddy 2012).

3 Experiment

We analyze the performance of our MCTS approach compared to two baselines.

Dataset. We generate random Erdős-Rényi compatiblity graphs with various sizes and sparsities. We converted these compatibility graphs into their respective MWIS graphs, and used Gurobi as our oracle to find an optimal solution.

Baselines. We compare our MCTS algorithm against two baselines, random and greedy. Random generates a random maximal independent set by picking a random action at each step of the MDP. Greedy repeatedly picks the node that minimizes the short term "loss", the difference between the node's weight and the weighted sum of the node's neighbors.

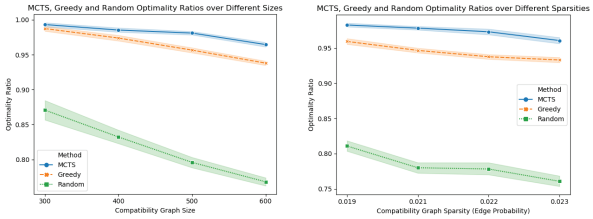

Figure 1: Optimality ratios of MCTS, Greedy, and Random on Erdős-Rényi graphs ($p = 0.02$), with 1 SD intervals.

Evaluation. We tested the performance of our algorithm and baselines by examining the optimality ratio, the ratio between the attained score and the optimal score.[2] The score is defined as the weighted sum of the nodes in the MWIS solution set, or the cardinality of the cycle cover.

Results. Our algorithm outperformed both baseline algorithms on a variety of compatibility graph sizes (Fig. 1).

4 Conclusion

We propose a novel method to approximately solve the combinatorial optimization problem of kidney exchange through GNN-guided MCTS. This is very early-stage, but promising, work (Fig. 1). Moving forward, we are applying some of the techniques described in the context of branch and bound search. Branch and bound enables us to utilize both the upper and lower bounds of a particular state, as well as the guidance from a learning method such as GNNs.

Acknowledgements. Dickerson was supported by NSF CAREER Award IIS-1846237 and a gift from Google.

References

Abraham, D.; Blum, A.; and Sandholm, T. 2007. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *EC*, 295–304.

Anderson, R.; Ashlagi, I.; Gamarnik, D.; and Roth, A. E. 2015. Finding long chains in kidney exchange using the traveling salesman problem. *PNAS* 112(3):663–668.

Barnhart, C.; Johnson, E. L.; Nemhauser, G. L.; Savelsbergh, M. W. P.; and Vance, P. H. 1998. Branch-and-price: Column generation for solving huge integer programs. *Operations Research* 46(3):316–329.

Browne, C. B., et al. 2012. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4(1):1–43.

Dickerson, J. P., and Sandholm, T. 2015. FutureMatch: Combining human value judgments and machine learning to match in dynamic environments. In *AAAI*, 622–628.

Sabharwal, A.; Samulowitz, H.; and Reddy, C. 2012. Guiding combinatorial optimization with uct. In *CPAIOR*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2019. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*.

[2] We solve to optimality the NP-hard optimization problem, formulated as a MIP, using Gurobi, and use this as an upper bound.