The Thirty-Third International
FLAIRS Conference (FLAIRS-33)

Decentralized Marriage Models

Kshitija Taywade, Judy Goldsmith, Brent Harrison

University of Kentucky
Lexington, Kentucky
kshitija.taywade@uky.edu, goldsmit@cs.uky.edu, harrison@cs.uky.edu

Abstract

Most matching algorithms are *centralized* in that a single agent determines how other agents are matched together. This is contrary to how humans form matches in the real world. We propose three decentralized approaches for finding matchings that are inspired by three techniques that humans use to find matches: a grid environment, with agents wandering around, interacting and deciding preferences over potential partners; affiliation networks where agencies recommend potential partners; and small-world social networks, where individuals are probabilistically introduced to one another by friends. We introduce a heuristic algorithm that can be used in each of these environments. We also explore how this algorithm can scale to a large number of agents.

Introduction

There are several personal, social, and cultural factors that influence how people find potential mates. In this paper, we explore the problem of finding optimal partner matchings under different assumptions about how people are introduced. We model the marriage problem as: a grid-world environment, a small-world network graph, and an affiliation network graph. Each of these environments represents a practically grounded way that people meet each other in the real world. The *grid-world* models agents that actively seek out potential partners on their own with little to no prior information about the other agents they will meet. In the *small-world network*, agents are presented with potential matches based on their degrees of separation. Thus, agents that are closer to each other in their social network are more likely to be considered as candidates for a match. *Affiliation networks* model the situation where agents are registered to matrimonial agencies and get potential matches suggested by those agencies. In many real world scenarios, it is not feasible for a centralized agency to optimally select partners for agents. Typically, agents — people — act to find partners autonomously without the need for a centralized agency. Thus, to better simulate real world scenarios we introduce a heuristic-based, decentralized approach for solving the marriage problem and show how it can be used, with slight modifications, to help agents find matches in each of the three environments described above. We show the effectiveness of our approach by demonstrating its ability to find good matchings in each of our three test models, where good matchings are defined as those that maximize the utility of all agents. We compare our results with several centralized and decentralized baselines. Our proposed approach can also be applied to other matching markets such as worker-employer and buyer-seller markets. As our method produces outcomes in polynomial time, it is fitting for environments/models where large number of agents need to make quick decisions.

Related Work

One popular set of approaches to solving marriage market problems — decentralized approaches — explore how random meetings can result in stable matches. Most of these approaches are focused on finding stable matches, and, like our work, use preferences or interests as the basis for their matchings (Vanzin and Barber 2006). Contrary to this work, our decentralized approach focuses on generating optimal matchings instead of stable matches.

Distributed algorithms for weighted matching mainly include algorithms that are distributed in terms of agents acting on their own either synchronously or asynchronously (Hoepman 2004; Wattenhofer and Wattenhofer 2004), and algorithms that are distributed using parallel programming to find approximate maximum matchings (Lotker, Patt-Shamir, and Rosén 2009; Manne and Bisseling 2007). In our work, we model matching problems in a grid-world and in social networks, and study the results produced by our decentralized approach.

Preliminaries

We model the marriage problem using bipartite matching, where S_1 and S_2 are the two vertex sets to be matched, and edges represent acceptable pairings.

Definition 1. *An instance of* Bipartite Matching *is a bipartite graph* $G = (S_1 \cup S_2, E)$ *with* $E \subseteq S_1 \times S_2$. *A solution is a maximal matching. We consider the problem of* Weighted Bipartite Matching, *where the edge weights correspond to preferences.*

Definition 2. *An instance of the* optimal marriage problem *is an instance of the weighted bipartite matching problem, where S_1 and S_2 correspond to the categories of agents, $p_{i,j}$ is the preference of i for j, i.e., the weight of edge (i, j), and the goal is to find a maximum weight matching. When preferences are asymmetric, the weight of a single pair is the sum of the weights of the two directed edges between nodes.*

Methods

In this section we introduce our models in more detail. We assume that agents start off agnostic about the world and about their potential mates in every model. We propose a decentralized approach to help autonomous agents find their partners in all the three models. The models differ in how, and how often, agents meet.

Grid-World Environment

Model The grid-world model is a spatial environment where agents can freely navigate the grid and encounter each other. We deploy agents at random cell locations on the grid. Agents can only see the contents of their current grid location.

Algorithm The grid-world environment contains S_1 and S_2 agents in equal numbers. At the beginning of each episode, agents are initialized at random starting locations. From here, they start randomly exploring the grid environment. When an agent encounters another agent of the opposite category, it discovers how much utility it can get if it is matched with that agent, and it stores that value if it is positive. It does not store the identity of the other agent. If two agents from the same category encounter each other, they ignore each other. Each agent has information about the total number of agents of the opposite category present in the environment. Using this, they can calculate how many potential matches could be formed and can, from that, reason about how many candidate agents they have not encountered yet. This helps determine if an agent forms a match and stays there, or if it chooses to explore to find a better match. We define the exploration rate r to be the ratio of steps so far to the total number of steps. For each individual, we define h to be the average of all positive utilities an agent has seen so far, c is the highest possible utility for an agent in the environment and u is the utility an agent gets from a match if it is part of that match. An agent decides whether to form a match with another agent when they encounter each other in the same cell, if they each get positive utility from the match. If an agent encounters more than one agent of the opposite category at the same cell then it chooses the best among them. If two agents are willing, then the match is formed. Agents only stay in the match when they find themselves in one of the following situations:

• If $u \geq 0.75 \times c$; • If $r \leq 0.6, u \geq h$;
• If $0.6 \leq r \leq 0.8, u \geq 0.5 \times h$; • If $r \geq 0.8, u \geq 0$.

In this way, agents first set high expectations and then gradually lower their expectations as time passes. Note that, if an agent encounters a better agent at the same cell while in a match, then it may leave the current match and form a new one with the better agent if that new agent is also interested. The matching at the end of the last episode is the resulting matching (the marriage). Each of these conditions can be tuned for specific environments.

Affiliation Network/Bipartite Network

Model It is usually used to represent common membership of groups, and in our case refers to membership in matrimonial agencies or dating websites. In this network, actors are connected by common membership. We represent such network as a bipartite graph, with nodes being either individuals (actors) or matrimonial agencies (collectives). Edges represent individuals' memberships in the agency. Matrimonial agencies suggest matches to their members. An actor can be a member of more than one agency.

Algorithm We create two types of agents, agencies and people, as nodes in the affiliation graph; people are randomly assigned to agencies. At each step, agencies randomly suggest a match to everyone who is registered with them. These suggestions are not necessarily symmetric: an agent may be suggested to another, without receiving the other as a suggestion. The one receiving a suggestion might express interest in the other. Each individual has a list of possible matches, including others they are interested in, and those who are interested in them. They may select one match to propose to, and be accepted or not. However, if they receive a better proposal in that time step, they retract their own proposal. If a proposal is offered and accepted and not retracted in that time step, then the marriage happens. When individuals form a match, they are removed from the agencies' registered candidate lists. If no marriage happens for an individual in a particular time step, then the individual's list is updated at the next step when they get new suggestions from agencies, until they get married. People are removed from others' lists when they get married. To recap: when an agent expresses interest in another agent, the couple is married unless the agent receives interest from a better candidate at the same time step, or the recipient refuses them. The factors that affect an agent's decisions on whether to propose, or to accept a proposal, are the utilities the agent has seen so far, and the time left. There are four possibilities listed below when agents decide to take actions (proposing to someone or accepting a proposal).

• If $r < 0.4$ & $c \geq 0.75 \times m$; • If $0.4 \leq r \leq 0.6$ & $c \geq h$;
• If $0.6 \leq r \leq 0.8$ & $c \geq 0.75 \times h$;
• If $0.8 \leq r$ & $c \geq 0.5 \times h$. If any of these conditions hold for an individual with respect to someone who was suggested, then they propose. If a proposal has been received, and the recipient finds themselves in one of those conditions, then they will accept the proposal. Note that, in the fourth case, expectations have been lowered, similarly to what happens as time runs out in the grid-world environment. Each of these conditions can be tuned for specific environments.

The Small-World Network

Model This is to formalize the process of finding matches through one's social network. In practice, people are often introduced by friends, or through chains of acquaintances. We randomly generate suggested matches, with probability inversely proportional to their distance in the graph, as a proxy for such introductions. Every node represents someone in search of a mate. We use a small-world network, where the average distance between nodes is somewhere be-

tween 6 and 7, based on the folkloric "six degrees of separation" introduced by Travers et al. (Travers and Milgram 1977). We use the Watts-Strogatz model (Watts and Strogatz 1998) of small-world graph construction to build our model.

Algorithm This is a variant of the Affiliation Network algorithm, where suggestions come not from agencies, but from the individuals' social network (small-world network). The likelihood of two individuals, i and j, meeting is a function of their distance $d(i, j)$ in the network. At each stage, individuals receive at most 3 suggestions/introductions. To generate the list of suggestions for i, the algorithm first samples (uniformly at random) one from each distance from i. The sampled individuals are added to the list with probability $\frac{1}{2d(i,j)}$. (Note that closer individuals are more likely to be added.) Finally, if the list has more than 3 suggestions, individuals on the list are chosen uniformly at random for culling from the list for this stage. We have chosen these parameters as they keep number of introductions per agent balanced, and similar to our other models. Note that unlike grid-world, in the affiliation network and social network algorithms, once agents become paired with someone they are considered married forever.

Experiments

We evaluate our algorithms on: quality, scalability, and robustness. To test quality, we calculate the total utility of all matchings found by our algorithm in each of the three models. In our models, agents try to increase their own utility, rather than global utility. We also examine the total number of matches found by each algorithm. We run each algorithm on each environment 10 times using different randomly assigned preferences between agents each time. The performance of our method is the average cumulative utility over these trials. To evaluate scalability, we perform these trials with 100 and 500 agents in each model. To evaluate the robustness of each algorithm with respect to observed utility values, we perform trials with utility ranges (1,10) and (-10,10). Thus, for each model we perform 4 sets of experiments: 100 agents, (1,10) utility range and (-10,10) utility range; 500 agents, (1,10) utility range and (-10,10) utility range. Our experiments consider both symmetric and asymmetric preferences for each set of trials run. Following are our comparison baselines:

Gale-Shapley (Gale and Shapley 1962).

Bidirectional Local Search attempts to find globally optimal stable matchings (Viet et al. 2016). Note that the goal of GS and BLS is to find stable matches, which is different than our goal.

Hoepman Algorithm is a decentralized variant of the sequential greedy algorithm which computes a weighted matching at least one half the maximum. (Hoepman 2004). This comparison is important because there, agents also act selfishly, as in our approaches.

Hungarian Algorithm finds optimal weighted bipartite matchings (Kuhn 1955). It is optimal in a utilitarian sense, meaning that individual agents may get low-utility matches in order to maximize the total utility, particularly in the assymmetric-preference case.

Decentralized Models and Algorithm Parameters

For each model, learning occurs in episodes in which agents take a fixed number of actions before episode concludes. We use two episodes to train each agent. Between episodes agents remember positive utilities they have received so far by being matched. The number of training steps in each episode depends on number of agents as well as the size of the grid: (1) 100 agents: $\big(20 \times 20\big)$ grid, 1000 steps (2) 500 agents: $\big(45 \times 45\big)$ grid, 30,000 steps. For affiliation networks, we use a bipartite version of the binomial (Erdös-Renyi) graph model (Erdös and Renyi 1959). To generate the graph, we specify the number of nodes in the first bipartite set (agents), n; the number of nodes in the second bipartite set (agencies), m; the probability for edge creation between agents and agencies, p. We use parameter combinations as: (1) n=100, m=5, p=0.5 (2) n=500, m=10, p=0.5. There is a 50% chance that an agent belongs to a given agency. In the resulting network, each agent is affiliated with 4–5 agencies on average. We chose these parameters by conducting an informal social survey amongst people that are registered to matrimonial agencies. To construct a small-world graph, we need: the number of nodes, n; the number of nearest neighbors that each node is joined with, k; and the probability of rewiring each edge, p. The average shortest path between any two nodes depends on these values. We need parameter values such that the average shortest path between any two nodes is about 6 (inspired by the *six degrees of separation* phenomenon). Therefore, for a network with 100 agents, we set n=100, k=5, and p=0.05, resulting in the average shortest path as approximately 6.2. For 500 agents, we have n=500, k=4, and p=0.15, resulting in average shortest path lengths of approximately 6.6.

Results and Discussion

We have compared our approach for the three marriage models to several centralized algorithms, and to the decentralized Hoepman algorithm. These results are summarized in Table 1. Our approaches significantly outperform the Hoepman algorithm in most cases, which highlights the power of our decentralized approach for solving matching problems when using a variety of representations. In addition, the fact that our method produced matchings with higher total utility despite each agent being selfish further shows the power of our algorithms. Moreover, our decentralized algorithms drastically outperformed the Gale-Shapley algorithm and gives quite close results to bidirectional local search algorithm in the set of experiments with asymmetric preferences. This is especially notable because our decentralized approaches are more generally applicable since they do not rely on a centralized agent. Despite that, our approaches were not able to outperform the Gale-Shapley algorithm in environments with symmetric preferences. This is likely due to how the Gale-Shapley algorithm finds matches. When preferences are symmetric, it is likely that the matchings produced by the algorithm are not so bad for the second group. Bidirectional local search algorithm is centralized, which is a big advantage that our approach does not have, therefore it performs better than our approach. We find that our models give some-

Table 1: Results in terms of average total utility averaged over 10 runs. We also list percentage of the optimal utility obtained. The highest performers among three models for each set of experiments are listed in bold.
Upper: For asymmetric preferences **Lower:** For symmetric preferences.

Agents; Utility Range	Gale-Shapley	Bidirectional Local Search	Hoepman Algorithm	Grid-world	Affiliation Network	Small-world Network	Optimal (Hungarian)
100; (1,10)	608.5 (65.1%)	915.2 (98.05%)	742.2 (79.51%)	**790.5 (84.69%)**	776.2 (83.14%)	753.3 (80.7%)	933.4
100; (-10,10)	134.6 (16.4%)	779 (95.10%)	676.9 (82.63%)	**678.1 (82.78%)**	592.5 (72.34%)	606.7 (74.06%)	819.1
500; (1,10)	2841.5 (57.30 %)	4881.7 (98.45 %)	3740.7 (75.44 %)	4018 (81.03 %)	**4368.1 (88.09 %)**	3924.2 (79.14 %)	4958.4
500; (-10,10)	461.5 (9.8%)	4577.3 (97.41 %)	3667 (78.04 %)	3821.9 (81.34%)	**3935.4 (83.76%)**	3626.8 (77.18%)	4698.6
100; (1,10)	943.4 (94.4%)	988 (98.89%)	738.8 (73.95%)	852.9 (85.37%)	843.2 (84.40%)	**875 (87.58%)**	999
100; (-10,10)	856.6 (87.8%)	937.4 (96.08 %)	729.6 (74.78 %)	799.3 (81.92%)	**817 (83.74%)**	814.2 (83.45%)	975.6
500; (1,10)	4943.6 (98.9%)	4995 (99.9 %)	3739 (74.78 %)	4332.1 (86.64%)	**4666.8 (93.33%)**	4553.6 (91.07%)	5000
500; (-10,10)	4827.2 (96.5%)	4974.6 (99.49 %)	3687.6 (73.75 %)	4237.6 (84.75%)	**4582.8 (91.65%)**	4486 (89.72%)	5000

what similar results. This roughly indicates that different environment settings do not significantly change the quality or the quantity of matches found. When we compare results for affiliation network and small-world network, we noticed that in 8 out of 10 different experiment settings, affiliation network performs better than small-world network setting. This may suggest that when agents are affiliated to agencies there is slightly better chance of finding a good partner than just depending on interpersonal networks, but it certainly needs more investigation as some of our parameters were based on our discretion and not on real-world data. The fact that each of our decentralized algorithms performed well across all environments and test cases indicates that they can be useful across different strategies for finding partners.

Conclusion

For proposed decentralized models and approach, the overall utility of matches found were, on average, 84% of the utility achieved by optimal (Hungarian) centralized algorithm. Since we argue that the social problem is inherently decentralized, agents have limited information, and typically preferences are asymmetric, so optimal sets of matchings are quite rare; our results are quite strong. Our approaches outperform the distributed Hoepman algorithm, which supports our claim. For asymmetric preferences, our algorithms performed significantly better that Gale-Shapley. However, when preferences are symmetric, the Gale-Shapley algorithm produced nearly optimal utility matchings, better than our approach. Millennia of literature assures us that human romantic preferences, however, are not symmetric.

References

Erdös, P., and Renyi, A. 1959. On random graphs I. *Publ. Math. Debrecen* 6:290–297.

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

Hoepman, J.-H. 2004. Simple distributed weighted matchings. *arXiv preprint cs/0410047*.

Kuhn, H. W. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2(1-2):83–97.

Lotker, Z.; Patt-Shamir, B.; and Rosén, A. 2009. Distributed approximate matching. *SIAM Journal on Computing* 39(2):445–460.

Manne, F., and Bisseling, R. H. 2007. A parallel approximation algorithm for the weighted maximum matching problem. In *International Conference on Parallel Processing and Applied Mathematics*, 708–717. Springer.

Travers, J., and Milgram, S. 1977. An experimental study of the small world problem. In *Social Networks*. Elsevier. 179–197.

Vanzin, M. M., and Barber, K. 2006. Decentralized partner finding in multi-agent systems. In *Coordination of Large-Scale Multiagent Systems*. Springer. 75–98.

Viet, H. H.; Lee, S.; Chung, T.; et al. 2016. A bidirectional local search for the stable marriage problem. In *2016 International Conference on Advanced Computing and Applications*, 18–24. IEEE.

Wattenhofer, M., and Wattenhofer, R. 2004. Distributed weighted matching. In *International Symposium on Distributed Computing*, 335–348. Springer.

Watts, D. J., and Strogatz, S. H. 1998. Collective dynamics of 'small-world' networks. *Nature* 393(6684):440.