The optimum matching policy for stochastic matching without feedbacks

Paper 224

Abstract

This paper studies a simplified version of the stochastic matching proposed in [Chen *et al.*, 2009]. The stochastic matching problem is modeled in a graph $G = (V, E)$. Each edge e_i has a probability to be existing. The market probs edges one by one (edge-probing setting). If an edge exists, it will be matched and both ends will leave the market. Otherwise, the edge can not be matched and one end of the edge will stay in the market only if she still has patience. The goal is to match more existing edges. Previous literature focus on finding bounds on settings with various parameters. However, these bounds are usually meaningless in simple settings. Even in many simple settings, finding the optimum policy to match meets great difficulties mainly from two aspects. First, there is no evidence that the optimal policy can be expressed in polynomial space. Second, collecting the probabilities usually incurs considerable costs.

We study a novel setting where the market does not know whether an edge exists after matching and the patience is defined in a new way. The goal in our setting is to find the optimum matching policy that maximizes the number of matched edges. We solve this problem via integer linear programming (ILP). Our well-designed ILP uses only $O(|V||E|)$ entries and outperforms the enumeration algorithm significantly. Although our work is for a quite specific setting, we move forward the first solid step towards "optimum" in the field of stochastic matching.

1 Introduction

Research on matching markets dates back to 1960s [Gale and Shapley, 1962], which has many applications, such as college admission, stable marriage problems and stable roommate problem [Roth, 1982]. In traditional matching settings, properties like stability, strategy-proof and social welfare maximizing are desired. However, during the last decade, inspired by applications such as kidney exchange [Roth *et al.*, 2003; 2005], adwords [Mehta *et al.*, 2007] and online dating [Hitsch *et al.*, 2010], more attention has been attracted to stochastic settings [Awasthi and Sandholm, 2009]. In such settings, stability and strategy-proof are hard to be well-defined. People care more about social welfare.

[Chen *et al.*, 2009] defines a model "stochastic matching" and starts an active line of study on worst case bounds on related problems[Adamczyk, 2011; Costello *et al.*, 2012; Poloczek and Szegedy, 2012; Goel and Tripathi, 2012; Bansal *et al.*, 2012]. They share some common assumptions in their models. The stochastic matching is modeled in a graph $G = (V, E)$. Each edge e has a probability to be existing. Otherwise, the edge is fake. There exists a matching system that can prob an edge to match. When an existing edge is probed, it confirms the match and will never be broken. When a fake edge is probed, it rejects the match and the edge is removed from the graph. The goal of the matching system is to maximize the number of matched edges or the total weight. Many variants have been studied in this setting, such as adding weights on vertices, probing an edge or a matching, setting the patiences, matching with eyes closed and matching in bipartite or general graphs[1]. All these study focus on finding bounds for the worst case compared to the optimum policy or the offline optimal solution. However, for the simple settings where all the existing probabilities are the same and weights are not considered, none of the algorithms above can be significantly better than serial dictatorship following a random order. The current settings face two main drawbacks that prevent it from practical use.

First, to know the exact probability of a prob being accepted is difficult. Typically, in order to know the exact existing probabilities, the market needs to collect many data and conduct careful analyses on historical data. It is worthy in some important markets like kidney exchange market [Dickerson *et al.*, 2013; Dickerson and Sandholm, 2015]. However, in many less important markets, collecting existing probabilities is quite costly and unworthy. Now, let us consider a dating problem. Someone (called Alice) has many friends, x of them are single boys and y are single girls. Alice wants to introduce boys to girls for marriage. She wants to maximize the number of matched couples. Every boy or girl has some requirements that must be satisfied. However, even though

[1] Often, more than one variants are considered in one paper at the same time, so we list the related work at the beginning of this paragraph.

a boy and a girl satisfy each other's requirements, they may still not be willing to match. In this case, it is hard and costly for Alice to survey the acceptance ratio (existing probability in the model) of each pair. What Alice does have may be the estimated average acceptance ratio of each pair. Such applications motivate us to study the settings where all the existing probabilities are unique. Expensiveness on surveying also appears in other applications like roommate problem, but never serious considered to the best of our knowledge.

Second, as mentioned above, most study focus on giving worst case bounds. The main difficulty preventing people to desire the optimum solution is that the optimum solution maybe not able to be expressed in polynomial space. Note that the optimum solution may not be implementable by serial dictatorship.

Motivated by the two drawbacks above, we study a novel setting with identical existing probabilities for edges and a simple expression for the number of expected matched agents. For short, we use the *payoff* of a matching to denote the number of expected matched agents. Our setting is inspired by the dating problem mentioned before. It is often the case that when a boy and a girl fall in love with each other, they will not tell it to Alice before finally deciding, but they will reject others. In our setting, the matching market probs one edge each time. The existing probabilities of edges are the same. We assume that the market cannot get feedbacks from the matched agents (vertices) in a short time because they are not willing to make it known to others. We assume that each agents has two status *active* or *inactive*. Typically, an agent becomes inactive because she has been tired of being probed or has been matched. We assume the probability of becoming inactive after each probing is a constant p. When an agent becomes inactive, she will always be inactive and rejects all the probs. When an edge between two active agents is probed, it will be matched with constant probability $r \leq p$ Our goal is to find the optimum matching policy. We find that the optimum matching policy is by maximum serial dictatorship and the optimum sequence for serial dictatorship can be solved by an integer linear program on an enlarged feasible space.

The following part of this paper is organized as follows. Section 2 defines the preliminaries and show some fundamental good properties of serial dictatorship. Section 3 shows the construction of the integer linear programming step by step and prove its correctness. Section 4 explores the extensions of our algorithm. Section 5 shows our experimental results. Section 6 concludes this paper.

2 The settings

In this section, we will describe our model and the serial dictatorship. After that, we will show that matching by serial dictatorship with the best sequence is the optimum policy.

2.1 The model

Our problem is modeled in an undirected graph $G = (V, E)$, $V = \{v_1, v_2, \ldots, v_n\}$ denotes the set of agents. $e_{ij} \in E$ denotes an edge between v_i and v_j. In the following analyses, elements in E are also renumbered as $e_1, e_2, \ldots, e_{|E|}$. For any vertex $v_i \in V$, it has two states, *active* and *inactive*. The initial states are active. When an edge e_{ij} is probed, v_i (or v_j) will choose one from the following actions.

- if v_i inactive, e_{ij} will not be matched.
- if v_i is active and v_j is inactive, v_i becomes inactive with probability p, e_{ij} will not be matched.
- if both v_i and v_j are active
  - with probability $r \leq p$, e_{ij} is matched and v_i becomes inactive.
  - with probability $p - r$, e_{ij} is not matched and v_i becomes inactive.
  - with probability $1 - p$, e_{ij} is not matched but v_i is still active.

In other words, we just assume that an agent has a probability p to become inactive after each probing and the existing ratios of edges are the same, denoted by r.

2.2 ESD algorithm

The framework of the *edge serial dictatorship algorithm* (ESD) is shown in Algorithm 1. It is quite natural and the performance of ESD depends on the sequence S.

Algorithm 1 ESD algorithm

Require: The graph $G = (V, E)$ and an ordered sequence of E, denoted by $S = s_1, s_2, \ldots, s_m$.
Ensure: A set of disjoint edges A, such that each two agents connected by an edge in A agree to match.

```
A = ∅
for i = 1, 2, ..., m do
    Let v_x and v_y be the two agents connected by s_i
    if v_x ∈ V and v_y ∈ V then
        Prob v_x and v_y whether they agree to match with each other.
        if Both v_x and v_y accept to match then
            Put s_i into A
            Remove v_x and v_y from V
        end if
    end if
end for
Outputs A as the set of matched edges.
```

Given the sequence S, The payoff of ESD algorithm is as follows.

$$obj(S) = \sum_{i=1}^{m} r(1-p)^{\sum_{j=1}^{i-1} \delta(s_j, s_i)} \tag{1}$$

In equation 1, $\delta(s_i, s_j)$ is an indicator of whether s_j appears before s_i and they have a common vertex. In other words, $\delta(s_j, s_i) = 1$ only when s_j is probed before s_i and they have a common vertex, otherwise $\delta(s_j, s_i) = 0$. In the integer linear programming, we will use variables δ_{ij}, which holds the same meaning as $\delta(s_i, s_j)$ here.

2.3 Optimum policy

The optimum matching policy in our setting is simpler to be found that other stochastic matching settings. The general idea is that because the market never gets any feedbacks, the action of the market is a deterministic sequence of probing.

Theorem 1 *In our setting, the payoff of the optimum policy is no better than the maximum payoff achieved by ESD.*

Proof: As the market does not get any feedback from the previous probing process, The only policy the market can apply is to prob according a sequence S_i from a set $\{S_1, S_2 \ldots, S_k\}$ with a certain probability pr_i. All the probabilities sum up to 1. The payoff is counted by weighted average. Obviously, $\sum_{i=1}^{k} pr_i * obj(S_i) \leq \max_{i=1}^{m} obj(S_i)$. In other words, the payoff of a combination is smaller than equal to the payoff achieved by the highest-payoff sequence in the set.

Now, we consider running ESP a single sequence, we are going to show that if the sequence can achieve the optimum payoff, it must contains all the edges. It's trivial because we can always put the unused edges in the last of the sequence to increase the payoff. ■

3 Optimizing the sequence

Algorithm 1 has given a framework of serial dictatorship. The problem is how to find the sequence S^* that maximizing the payoff. In this section, we will introduce the integer linear program (ILP) for finding the optimum S^* step by step.

3.1 Overview of the algorithm

The objective function of our ILP is a representation of Equation 1. As for the constraints, if we faithfully write the constrains according to the definition of serial dictatorship and the objective function, we need $O(|E|^4)$ variables in the ILP, which is insufferable. We are not going to discuss this idea in details.

To reduce the number of variables, we introduce a novel method. We first enlarge the solution space to obtain a relaxed-form ILP. Then, we show that the solution of the relaxed-form ILP can be mapped to the optimal sequence S^*.

The relaxed-form ILP is based on the following observation. Given the optimal sequence S^*, there is a determined order between any two edge. For example, for $e_{ij}, e_{ik} \in E$, given S, we know that e_{ij} is always probed before e_{ik} or e_{ik} is always before e_{ij}. Now, we are ready to enlarge the solution space. Rather than a total order, we only assume that there is a determined order between any two edges with a common vertex (aka. *neighboring edges*). We do not have any other constraints on the solution space. Then, we will introduce a method that maps the optimal solution of the relaxed-form ILP to the optimal sequence S^*. The relaxed-form ILP only has $O(|E||V|)$ variables and $O(|E||V|)$ entries in total. So, it improves the performance significantly.

3.2 Variable Definition

Three types of variables are used in our ILP (from here on, all the "ILP" refers to the relaxed-form ILP). For a better presentation of our ILP, we give an alias for each element in E. We number all the edges in E as $e_1, e_2, \ldots, e_m$. This notation is not conflict with the previous definition. An edge e_i may denote the same edge as e_{jk}. Now, we are ready to define the variables in our setting.

- $\delta_{ij}(i, j \in \{1, 2, \ldots, m\})$ denotes an indicator of e_i is probed before e_j and they have a common vertex (just as mentioned before). If both conditions are satisfied $\delta_{ij} = 1$, otherwise $\delta_{ij} = 0$. The value of δ_{ij} is selected from $\{0, 1\}$. Actually, we can define δ only on neighboring edge pairs to reduce the number of variables.
- $o_i(i \in \{1, 2, \ldots, m\})$ denotes the number of edges that are probed prior to e_i and have a common vertex as e_i. By definition, we have $o_i = \sum_{j=1}^{m} \delta_{ji}$. This equation is also used as a constraint in the ILP. The value of o_i is selected from nonnegative integers
- $l_{ij}(i, j \in \{1, 2, \ldots, m\})$ denotes an indicator for whether o_i is greater than or equal to j. This variable is used to express the objective function of ILP. Its value is selected from $\{0, 1\}$.

3.3 Objective function

The objective of the ILP is to maximize to total number of cardinality, which has been shown in Equation 1. We write it in ILP fashion as follows.

$$
\begin{aligned}
obj &= \sum_{i=1}^{m} r * (1-p)^{o_i} && (2)\\
&= r * \sum_{i=1}^{m} ((1-p)^{o_i} - (1-p)^{o_i - 1} + (1-p)^{o_i - 1} - \\
&\quad (1-p)^{o_i - 2} + \ldots - (1-p) + (1-p) - 1 + 1) \\
&= mr + r \sum_{i=1}^{m} \sum_{j=1}^{o_i} ((1-p)^j - (1-p)^{j-1}) \\
&= mr + r \sum_{i=1}^{m} \sum_{j=1}^{m} ((1-p)^j - (1-p)^{j-1}) l_{ij} && (3)
\end{aligned}
$$

In equation 3, the objective function obj becomes a linear function, such that it can be optimized by the ILP. The current problem is how to make l_{ij} work as an indicator for $o_i \geq j$.

As stated before, the solution of the ILP only makes sense when the objective function obj has been maximized. So, we only need to add the following constraint to restrict l_{ij}.

$$o_i \leq j - 1 + l_{ij} * |E| \quad (4)$$

In our ILP, l_{ij} only appears in the constraint above and the objective function.

Lemma 1 *When obj has been maximized, $o_i \geq j$ if and only if $l_{ij} = 1$.*

Proof: To prove this lemma, we only need to show that the following two cases are impossible when obj is maximized.

- $o_i < j$ and $l_{ij} = 1$.
- $o_i > j$ and $l_{ij} = 0$.

The second case is obviously impossible by Equation 4.

If the first case is satisfied and obj is maximized, set l_{ij} to be 0 will increase the obj and all the constraints are still satisfied. A contradiction. ■

3.4 The ILP

Now we are ready to conclude the ILP we used to solve the optimal serial as follows.

$$\text{Maximize:} \quad mr + r\sum_{i=1}^{m}\sum_{j=1}^{n_i}((1-p)^j - (1-p)^{j-1})l_{ij} \quad (5)$$

$$\text{Subject to:} \quad 1.\ \delta_{ij} + \delta_{ji} = 1, \forall e_i, e_j \in E, e_i \cap e_j \neq \emptyset \quad (6)$$

$$2.\ o_i \leq j - 1 + l_{ij}|E|, \quad i = 1, 2, \ldots, m, j = 1, 2, , \ldots, n_i \quad (7)$$

$$3.\ o_i = \sum_{e_j \cap e_i \neq \emptyset, i \neq j} \delta_{ji}, \forall i = 1, 2, \ldots, m \quad (8)$$

$e_i \cap e_j \neq \emptyset$ denotes that e_i and e_j are neighboring edges. In equation 7 and the objective function, n_i denotes the number of edges that have a common vertex as e_i. Introducing n_i decreases the number of variables we used. Further, we omitted the δ_{ij}'s for disjoint e_i and e_j. These two steps ensure the number of entries to be $O(|E||V|)$. We will discuss the complexity problem later.

3.5 Mapping to the optimum sequence

We note that when the objective function is not maximized, a feasible solution of the ILP may not correspond to a feasible sequence for serial dictatorship. However, when the objective function is maximized, the optimum solution of the ILP can be mapped to a feasible sequence for serial dictatorship as follows.

Mapping process: First, we define a function o satisfying $o(e_i) = o_i^*$. Note that we reuse the o because of the same meaning. o_i^* denotes the value of o_i when the objective function in the ILP has been maximized. Then, we sort E to be a sequence $S^* = s_1^*, s_2^*, \ldots, s_m^*$, satisfying $\forall i, j \in \{1, 2, \ldots, m\}, i < j$, $o(s_i) \leq o(s_j)$. The S^* is the optimal sequence for ESP.

3.6 A summary of our algorithm

For a better understanding, our algorithm for optimum serial dictatorship is shown in Algorithm 2. The overall idea is to solve the ILP, map the optimum solution to a sequence and run ESD algorithm.

3.7 The correctness proof

Recall the definition of o_i, we just sort all the edges according to the number of neighboring edges probed before the edge. For short, we call "the number of neighboring edges probed before the edge" as the *order* of the edge. Now, we are going to demonstrate that S^* is the optimal sequence. The rough idea is that the optimal solution of ILP is always implementable by ESD. The reason is that an edge always probed earlier than one with higher order. Otherwise, let lower-order edge be probed first can increase the value of the objective function.

Formally, it is stated as follows.

Algorithm 2 Optimum serial dictatorship

Require: A graph $G = (V, E)$

Ensure: A set of disjoint edges A, such that each two agents connected by an edge in A agree to match and the expected size of A is maximized.

1: Run the ILP shown in Equations 5-8 on graph G. The value profile for the optimum solution is denoted by P.
2: Map P to a sequence S^* by mapping process.
3: Run Algorithm 1 on S^*, outputs the matched edges A as output.

Theorem 2 *The output sequence S^* of our algorithm is the sequence that maximizes the payoff in the ESD algorithm.*

Proof: The proof of this theorem consists of two parts. Case (1): We show that any sequence of edges corresponds to a point in the ILP's solution space. This property indicates that the maximum value of the ILP is no smaller than the maximum expected number of matched edges. Case (2): We show that the sequence S^* constructed by the mapping process corresponds to the optimal solution of the ILP. In other words, the payoff of ESD with sequence S^* is the same as the maximum value of the objective function. It indicates that the maximum value of the ILP is no bigger than the maximum expected number of matched edges.

Case (1): Given a sequence of edges $S = s_1, s_2, \ldots, s_m$, we can construct the variables in the ILP as follows. For any pair of neighboring edges e_i and e_j with $i < j$. $\delta_{ij} = 1$, otherwise $\delta_{ij} = 0$. According to the ILP, $\delta_{ji} = 1 - \delta ij$. Let o_i equal to $\sum_{j=1}^{m} \delta_{ji}$. $l_{ij} = 1$ if and only if $o_i \geq j$. We call this process as *reverse mapping process*. In this way, we have map S to a point in the solution space of the ILP. Further, the value of objective function under this construction is exactly the expected number of matched edges. So we conclude the maximum objective function value is greater than or equal to the maximum expected number of matched edges.

Case (2): We are going to show that S^* corresponds to the optimum solution of the ILP by reverse mapping. To prove this, we only need to prove that the values of all the variables are the same for the following two cases:

- the optimum ILP solution where S^* is mapped from;
- the variables generated by reverse mapping from S^*.

Then, we show that we only need to compare the values of δ_{ij}'s.

- In the optimal solution, according to Lemma 1, l_{ij}'s are decided by o_i's. Further, o_i's are decided by δ_{ij}'s according to the equation 8. So, when δ_{ij}'s are fixed in the optimal solution, the values of all the other variables are fixed.
- For the variables generated by reverse mapping from S^*, l_{ij}'s, o_i's and δ_{ij}'s follow the same relationship as the former case. In details, $l_{ij} = 1$ if and only if $o_i \geq j$, and $o_i = \sum_{j=1}^{m} \delta_{ji}$. So if the two cases have the same δ_{ij}'s, all the variables are the same.

To prove that the two cases have the same δ_{ij}'s, we need the following lemma.

Lemma 2 *In the optimum solution of ILP, if $o_j \leq o_i$, $\delta_{ij} = 0$*

Proof: We prove this lemma by contradiction. By contradiction, we assume that in the optimum solution, there exist i and j satisfying $o_j \leq o_i$ and $\delta_{ij} = 1$. By Lemma 1 and Equations 2 to 3, in the optimum solution, the Equation 2 is equivalent to the objective function of the ILP. So, in the optimum solution, the value of the objective function is as follows.

$$obj_1 = \sum_{i=1}^{m} r(1-p)^{o_i} \tag{9}$$

Then, let $\delta_{ij} = 0$ and $\delta_{ji} = 1$ and keep the other δ's unchanged. We denote the new values of o_i and o_j by o'_i and o'_j. We have $o'_j = o_j - 1$ and $o'_i = o_i + 1$. Now, we consider the point in the feasible space where $l_{ij} = 1$ if and only if $o_i \geq j$. The value of the objective function at this point is as follows.

$$\begin{aligned} obj_2 &= obj_1 + p(1-p)^{o_j-1} + p(1-p)^{o_i+1} \\ &\quad -p(1-p)^{o_j} - p(1-p)^{o_i} \end{aligned} \tag{10}$$

Thus, we have:

$$\begin{aligned} obj_2 - obj_1 &= p(1-p)^{o_j-1} + p(1-p)^{o_i+1} \\ &\quad -p(1-p)^{o_j} - p(1-p)^{o_i} \\ &= p(1-p)^{o_j-1}(1 + (1-p)^{o_i-o_j+2} \\ &\quad -(1-p) - (1-p)^{o_i-o_j+1}) \\ &> 0 \end{aligned}$$

So, we find a feasible point whose objective function's value is larger than optimum. A contradiction. ■ Now, we consider a pair of edges e_i and e_j.

- If $\delta_{ij} = 1$, by Lemma 2, $o_i < o_j$. Then, in S^*, e_i is ahead of e_j. Then, after reverse mapping, as e_i and e_j have a common vertex and e_i is ahead of e_j, $\delta_{ij} = 1$.
- If e_i and e_j are disjoint, after mapping and reverse mapping, $\delta_{ij} = 0$
- If $\delta_{ij} = 0$ but e_i and e_j have a common vertex, e_j is ahead of e_i in S^*. After reverse mapping, δ_{ij} still equals to 0.

So, all the δ_{ij}'s are the same between the optimum solution where S^* is generated from and the feasible point S^* corresponds to in the ILP. Above all, we have finished the proof for the second case. ■

3.8 Complexity analysis

Even though solving ILP is an NP-Hard problem, instances at a small scale is solvable by many ILP solvers. We care about the number of entries in our ILP. By direct observations, we find the number of entries can be bounded by $O(|E|^2)$. However, with carefully analysis, we reduce it to $O(|V||E|)$.

Theorem 3 *The ILP only has $O(|V||E|)$ entries.*

Proof: In the ILP, each δ_{ij}, o_i or l_{ij} appears no more than 2 times. So, we only need to show that the number of variables is $O(|V||E|)$.

- The number of o_i's is $|E|$.
- For each pair of neighboring edges e_i and e_j, they are two corresponding variables δ_{ij} and δ_{ji}. The total number of pairs of neighboring edges can be counted as follows. For an arbitrary edge e_i if e_j and e_j has a common vertex. The common vertex has two possibles and the other end of e_j has $|E| - 2$ possibles. So, the number of δ's is $O(|V||E|)$.
- The number of l_{ij}'s is exactly the sum of n_i's. Each pair of neighboring edges increases $\sum_{i=1}^{m} n_i$ by 2. So, the number of l_{ij}'s is $O(|V||E|)$.

■

4 Extensions of our algorithm

Our algorithm can perform well on small graphs. However, for larger graphs, ILP is too slow. As our algorithm is the fastest way to get the optimum sequence for serial dictatorship to the best of our knowledge, we need find methods to approximate the optimum sequence for larger graphs.

The optimum solution in our setting is also related to a basic setting of stochastic matching. In that setting, even the payoff of serial dictatorship can not be counted easily. Our objective function can be used as a lower bound for serial dictatorship in that setting.

4.1 Force mapping

When the size of the graph is big enough, the ILP can not be solved in an acceptable time limit. However, our mapping process can always get a feasible solution.

The rough idea of the force mapping is as follows. Force mapping is based on Algorithm 2. The only difference is that we terminate the ILP at a fixed time T. At that time, the *variable profile* (the values of all the variables) is denoted by P and the corresponding objective value is denoted by obj. Then, map P to a sequence S by the mapping process. S is used as the sequence for ESD algorithm.

The sequence generated by force mapping does not have any guarantee. However, our experiments will show that force mapping can yield good payoffs.

4.2 Relation to a basic stochastic matching setting

In our setting, we have two major adjustments on the traditional setting to make the optimum policy solvable at a certain degree. First, we assume that the market does not get any feedbacks. This assumption restricts the optimum policy to be non-adaptive. Second, we assume that after one agent is probed, she has a fixed probability to leave the market. This assumption reduce the difficulty in expressing the payoff of a policy. However, our work can also contribute to the solving of traditional settings.

Now, we consider a simple version of stochastic matching. We use *basic setting* to denote this setting. This setting is also built on a graph $G = (V, E)$. The market probs edges one by one. When one edge is probed, it has probability p to be matched. Otherwise, this edge will be removed from the graph. A vertex will never be inactive unless it has been matched. Given a sequence $S = s_1, s_2, \ldots, s_m$. We use $N(i)$ to denote the set of edges that have common vertices with s_i

Table 1: Running time comparison between Algorithm 2 and the naive algorithm.

n,m,algorithm	maximum	minimum	median	mean
6,7,ILP	0.110	0.047	0.047	0.057
6,7,naive	0.063	0.046	0.062	0.055
6,8,ILP	0.131	0.062	0.099	0.099
6,8,naive	0.587	0.565	0.575	0.575
6,9,ILP	0.245	0.078	0.128	0.130
6,9,naive	6.467	6.271	6.353	6.349

and are probed prior to s_i. Then the probability of s_i gets matched, pr_i is as follows.

$$pr_i = p \prod_{j \in N(i)} (1 - pr_j) \tag{11}$$

Then, the payoff of the serial dictatorship on S is:

$$obj'(S) = \sum_{i=1}^{m} pr_i \tag{12}$$

We note that in the Equation 11, pr_i's depends on each other. However, for any i, pr_i is bounded by p. So the objective function of our setting $obj(S)$ when $r = p$ is a lower bound for $obj'(S)$. By optimizing the objective function in our setting, we can ensure the payoff of the basic setting to be not so bad.

5 Experiments

In our experiments, we test Algorithm 2 on both computational time and payoff. Algorithm 2 can deal with a graph with $|V||E|$ up to 200 and its payoff (optimum) is much better than simple algorithms.

5.1 Experimental settings

All our experiments are conducted on a common PC with a quad-core CPU and 4G RAM. We use CPLEX as our ILP solver.

As the running times of all the algorithms are not significantly relied on the leaving probability p (we confirm this by some experiments), we set p to be 0.7. As for the probability of accepting a prob r, it definitely does not affect running time. We just set $r = p = 0.7$.

5.2 Scalability

As there is no previous literature on finding the optimum sequence in our setting, we compare our algorithm to the most naive algorithm, which enumerates all the permutations of the edges and finds the best of them.

Table 1 shows the running time of Algorithm 2 and the naive algorithm for three cases. For each case, we test 20 instances with n as the number of vertices and m as the number of edges. "ILP" and "naive" stands for the two algorithms. The unit of all the numbers in the table is seconds. As the running time of the naive algorithm is $\Theta(|E|! \times |E|^2)$. It can be concluded that when the number of edges is greater than 7, Algorithm is significantly faster. When $|E| = 7$, the result seems to be a little bit wired. However, it is reasonable because the graph is too sparse and many instances may have similar structures.

Table 2: Running time of Algorithm 2 for larger instances.

n,m	maximum	minimum	median	mean
10,14	0.781	0.157	0.234	0.314
10,16	9.510	0.281	0.664	1.864
10,18	19.157	2.337	3.447	5.162
10,20	65.872	13.830	17.763	22.424
10,21	523.552	25.045	62.312	101.141

Table 3: Payoff comparison between algorithms

$\|E\|$	14	16	18	20
random	2.698	2.702	2.702	2.636
greedy	2.988	2.960	2.869	2.757
optimum	3.593	3.691	3.738	3.801

Table 2 shows the performance of Algorithm 2 on larger cases. We find that the worst case running time of our algorithm increase dramatically. We set $n = 10$, because only under this condition, the graphs with $|V||E| \approx 200$ is neither too sparse nor too dense. Under such condition, our algorithm can often solve instances with $m = 20$ within 1 minutes. Although the scalability of our algorithm is far from perfect., it has been much better than enumeration with the help of a highly non-trivial ILP.

5.3 Payoff comparison

In our setting, Algorithm 2 is able to generate the optimum sequence. To show the importance of the "optimum", we compare our algorithms with some other methods for generating the sequence. We use the following two algorithms as our baselines.

- Random algorithm: Run ESD on a random sequence.
- Greedy algorithm: sort all the edges according to the total degrees of their two ends from low to high. Use the sorted sequence as the input of the ESD algorithm. Intuitively, this is motivated by the observation that an edge tends to have less bad effects on others if the total degree of its two ends is small.

We test our algorithm and the two baselines on 4 cases, where $|V| = 10$ and $|E| = 14, 16, 18, 20$.

Table 3 shows the comparison between the three algorithms. From the results, we conclude that the optimum payoff (expected matched edges) is significantly better than the two baselines. The greedy algorithm becomes worse as the number of edges increase. The random algorithm is always the worst among these three algorithms.

6 Conclusion

In the field of stochastic matching, previous study concentrates on worst-case bounds, rather than "optimum". We simplify the original setting by forbidding feedbacks and changing the assumption on patience, making it possible to find a more efficient way to get the optimum policy. Our setting is motivated by the dating problem. In our setting, the optimum matching policy can be proved to be serial dictatorship, whose optimum sequence can be solved by a well-designed ILP. Our algorithm performs much faster than enumeration in our setting. Further, we compare our algorithm with some other algorithms, showing the importance of the optimal solution.

References

[Adamczyk, 2011] Marek Adamczyk. Improved analysis of the greedy algorithm for stochastic matching. *Information Processing Letters*, 111(15):731–737, 2011.

[Awasthi and Sandholm, 2009] Pranjal Awasthi and Tuomas Sandholm. Online stochastic optimization in the large: Application to kidney exchange. In *IJCAI*, volume 9, pages 405–411, 2009.

[Bansal *et al.*, 2012] Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When lp is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762, 2012.

[Chen *et al.*, 2009] Ning Chen, Nicole Immorlica, Anna R Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. *Automata, Languages and Programming*, pages 266–278, 2009.

[Costello *et al.*, 2012] Kevin P Costello, Prasad Tetali, and Pushkar Tripathi. Stochastic matching with commitment. In *Automata, Languages, and Programming*, pages 822–833. Springer, 2012.

[Dickerson and Sandholm, 2015] John P Dickerson and Tuomas Sandholm. Futurematch: Combining human value judgments and machine learning to match in dynamic environments. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

[Dickerson *et al.*, 2013] John P Dickerson, Ariel D Procaccia, and Tuomas Sandholm. Failure-aware kidney exchange. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, pages 323–340. ACM, 2013.

[Gale and Shapley, 1962] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *American mathematical monthly*, pages 9–15, 1962.

[Goel and Tripathi, 2012] Gagan Goel and Pushkar Tripathi. Matching with our eyes closed. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 718–727. IEEE, 2012.

[Hitsch *et al.*, 2010] Günter J Hitsch, Ali Hortaçsu, and Dan Ariely. Matching and sorting in online dating. *The American Economic Review*, pages 130–163, 2010.

[Mehta *et al.*, 2007] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22, 2007.

[Poloczek and Szegedy, 2012] Matthias Poloczek and Mario Szegedy. Randomized greedy algorithms for the maximum matching problem with new analysis. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 708–717. IEEE, 2012.

[Roth *et al.*, 2003] Alvin E Roth, Tayfun Sonmez, and M Utku Unver. Kidney exchange. Technical report, National Bureau of Economic Research, 2003.

[Roth *et al.*, 2005] Alvin E Roth, Tayfun Sönmez, and M Utku Ünver. Pairwise kidney exchange. *Journal of Economic theory*, 125(2):151–188, 2005.

[Roth, 1982] Alvin E Roth. Incentive compatibility in a market with indivisible goods. *Economics letters*, 9(2):127–132, 1982.