Resource Allocation to Agents with Restrictions: Maximizing Likelihood with Minimum Compromise

Yohai Trabelsi[1], Abhijin Adiga[2], Sarit Kraus[1], and S. S. Ravi[2,3]

[1] Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
yohai.trabelsi@gmail.com, sarit@cs.biu.ac.il
[2] Biocomplexity Institute and Initiative, Univ. of Virginia, Charlottesville, VA, USA
abhijin@virginia.edu
[3] Dept. of Computer Science, University at Albany – SUNY, Albany, NY, USA
ssravi0@gmail.com

Abstract. Many scenarios where agents with restrictions compete for resources can be cast as maximum matching problems on bipartite graphs. Our focus is on resource allocation problems where agents may have restrictions that make them incompatible with some resources. We assume that a Principal chooses a maximum matching randomly so that each agent is matched to a resource with some probability. Agents would like to improve their chances of being matched by modifying their restrictions within certain limits. The Principal's goal is to advise an unsatisfied agent to relax its restrictions so that the total cost of relaxation is within a budget (chosen by the agent) and the increase in the probability of being assigned a resource is maximized. We establish hardness results for some variants of this budget-constrained maximization problem and present algorithmic results for other variants. We experimentally evaluate our methods on synthetic datasets as well as on two novel real-world datasets: a vacation activities dataset and a classrooms dataset.

Keywords: Matching advice, Bipartite matching, Resource allocation, Submodular function

1 Introduction

There are many practical contexts where a set of **agents** must be suitably matched with a set of **resources**. Examples of such contexts include matching classes with classrooms [24], medical students with hospitals [27], matching buyers with products [19], matching customers with taxicabs [12], matching agricultural equipment with farms [13,26], etc. We assume that the matching process assigns at most one resource to each agent and that each resource is assigned to at most one agent. It is possible that some agents are not assigned resources and some resources are unused.

Agents have **restrictions** (or preferences) while resources have **constraints**. We assume that agents' restrictions are *soft*; that is, agents are willing to *relax*

their restrictions so that they can get a resource. An agent who is unwilling to compromise may not get any resource. However, the constraints associated with resources are *hard*; they *cannot* be relaxed.

Example: An instructor who indicates her restriction for the classroom capacity as "Capacity $\geq$ 70" may be willing to relax this restriction to "Capacity $\geq$ 60" to improve her chances of obtaining a classroom. However, a classroom of size 50 imposes the hard constraint "Capacity $\leq$ 50".

An agent is **compatible** with a resource (i.e., the agent can be matched with the resource) only when the (hard) constraints of the resource are satisfied by the agent's restrictions. The problem of assigning resources to agents can be modeled as a matching problem on the following bipartite graph, which we refer to as the **compatibility graph**: the graph has two disjoint sets of nodes corresponding to the agents and resources respectively; each edge $\{u, v\}$ in the graph indicates that the agent represented by u is compatible with the resource represented by v. A Principal (who is not one of the agents) chooses a maximum matching in the graph to maximize the number of agents who are assigned resources. Usually, there are many such maximum matchings, each one allocating resources to a (possibly) different set of agents. For fairness, the Principal chooses a maximum matching randomly out of a given distribution. The Principal may use, for example, an algorithm for fair matching [10] or a straight-forward process that randomly orders the agents and uses a deterministic matching algorithm like the Hopcroft-Karp algorithm [15] to generate a maximum matching.

It is natural for an agent, who is concerned that she will not be matched in the randomly generated matching, to seek advice from the Principal in the form of changes to her restrictions in order to increase the likelihood of getting matched. We assume a nonnegative cost associated with relaxing each restriction. Agents are desperate to get such advice when there are several rounds of matching and they failed in previous ones; such a situation arises, for example, in the case of medical students who were not matched during the first round of the residency matching process [17]. Developing such recommendations can be modeled as the following budget-constrained optimization problem: find a set of modifications to an unmatched agent's restrictions under a budget constraint so that the likelihood of the agent being matched to a resource is maximized, given the resource compatibility information for the other agents.

Several recommendation systems in environments where agents compete for resources are similar to our notion of a Principal. As an example, many route planning and satellite navigation apps provide advice to a given agent (driver) without taking into account possible changes in the behaviors of other agents due to similar recommendations. These recommendations often lead to undesirable consequences that are referred to as the price of anarchy [30]. The study of how to decrease the price of anarchy is beyond the scope of this paper.

Summary of contributions.

1. The matching advice problem. We develop a formal framework for advising agents in a resource allocation setting viewed as a matching problem on an agent-resource bipartite graph. We formulate a budget-constrained optimization

problem to generate suitable relaxations of an unmatched agent's restrictions so as to maximally increase the probability that the agent will be matched. We identify and study different forms of restrictions arising from agent restrictions and resource properties in real-world applications.

2. Complexity of improving the likelihood of matching. We show that, in general, the budget-constrained optimization problem is **NP**-hard.

3. Algorithms for improving the likelihood of matching. Under uniform costs for relaxing restrictions and uniform random selection of maximum matchings, we present algorithmic results for some classes of restrictions (which will be defined in Section 2.3). Specifically, we present an efficient approximation algorithm (with a performance guarantee of $(1 - 1/e)$) for the Multi-Choice Single-Restriction case. This result relies on the submodularity of the objective function. For another class called *threshold-like* restrictions, we develop a fixed parameter tractable algorithm, assuming that the budget and the cost of removing each restriction are non-negative integers.

4. Experimental Study. We study the performance of our recommendation algorithms on both synthetic data sets as well as two real-world data sets. The latter data sets arise in the contexts of assigning classrooms to courses and matching children with activities. We evaluate our algorithms under different cost schemes. The insights gained from this study can inform the PRINCIPAL (e.g., university administration) on issues such as adding, removing or modifying resources to cater to the needs of agents.

Related work. Resource allocation in multi-agent systems has been studied by a number of researchers (e.g., [3,5,14]). The general focus of this work is on topics such as how agents express their resource requirements, algorithms for allocating resources to satisfy those requirements and evaluating the quality of the resulting allocations. Nguyen et al. [21] discuss some complexity and approximability in this context [21]. Zahedi et al. [31] study the problem of allocating tasks to agents in such a way that the task allocator can respond to queries dealing with counterfactual allocations.

Motivated by e-commerce applications, Zanker et al. [32] discuss the design and evaluation of constraint-based recommendation systems that allow users to specify soft constraints regarding products of interest. These constraints are in the form of rank ordering of desired products. Both algorithms for the problem and a system which includes implementations of those algorithms are discussed in [32]. Felfernig et al. [9] provide a discussion on the design of constraint-based recommendation systems and the technologies that are useful in developing such systems. Parameswaran et al. [23] discuss the development of a recommendation system that allows university students to choose courses; the system has the capability to handle complex constraints specified by students as well as those imposed by courses. Zhou and Han [33] propose an approach for a graph-based recommendation system that groups together agents with similar restrictions to allocate resources. To our knowledge, the problem studied in our paper, namely advising agents to modify their restrictions to improve their chances of obtaining resources, has not been addressed in the literature.

Note: Proofs of many results mentioned in the paper appear in the appendix.

2 The Matching Advice Framework

2.1 Graph Representation and Problem Formulation

Agents, resources, and compatibility. We consider scenarios consisting of a set of *agents* (denoted by $\mathbb{X}$) and a set of *resources* (denoted by $\mathbb{Y}$). Every agent would like to be matched to a resource. However, agents may have *restrictions* that prevent them from being matched to certain resources. Such agent-resource pairs are said to be *incompatible*. We represent this agent-resource relationship using an $\mathbb{X}\mathbb{Y}$-bipartite graph called the *compatibility graph* $G(\mathbb{X}, \mathbb{Y}, E)$, where the edge $\{x, y\} \in E(G)$ iff the agent $x \in \mathbb{X}$ is compatible with $y \in \mathbb{Y}$. A PRINCIPAL assigns resources to agents. To maximize resource usage, the PRINCIPAL picks a *maximum matching* [4] from the compatibility graph.

The advice seeking agent and its restrictions. The special agent who seeks advice will henceforth be denoted by x^*. Let $\mathbb{Y}_I \subseteq \mathbb{Y}$ be the set of resources that are incompatible with x^*. Let $R = \{r_1, r_2, \ldots, r_\ell\}$ be the set of restrictions of x^*. A *resource–restrictions* pair (y, R') consists of a resource y and a restriction set $R' \subseteq R$ such that (i) y is incompatible with x^* and (ii) R' is a minimal set of restrictions to remove so that y becomes compatible with x^*. A resource-restriction pair describes precisely why resource y is currently incompatible with x^* (i.e., the edge $\{x^*, y\}$ is not in the compatibility graph), and how it can be made compatible. Suppose a set A of restrictions is removed. Then, a previously incompatible resource y becomes compatible iff there exists $(y, R') \in \Gamma$ such that $R' \subseteq A$. We then add the new edge $\{x^*, y\}$ to the compatibility graph. Let $\Gamma = \{(y, R') \mid y \in \mathbb{Y},\ R' \subseteq R\}$ be the set of such resource–restrictions pairs. We refer to Γ as the *incompatibility set* of x^*. Note that there could be more than one resource–restrictions pair with the same resource when there are multiple choices for removing restrictions to make the resource compatible with x^*. For a restriction $r \in R$, let $\rho(r)$ be a positive real number denoting the cost incurred by x^* for relaxing r. For any $A \subseteq R$, the cost of relaxing all the restrictions in A is $\rho(A) = \sum_{r \in A} \rho(r)$.

Resource allocation using bipartite maximum matching. To maximize resource usage and ensure fairness for all agents, we assume that the PRINCIPAL picks a maximum matching from the set of all possible maximum matchings. There are two components to this part of the framework: (i) generating a random maximum matching of the compatibility graph and (ii) computing the probability that x^* is picked in a random maximum matching. Firstly, we note that a maximum matching of a bipartite graph can be obtained in polynomial time [15]. Given any deterministic algorithm for maximum matching, one can permute the set of agents or resources (or both) randomly and obtain a random matching or one could use approaches such as the fair matching algorithm [10]. In any case, the first part can be computed in polynomial time. The second part however is computationally harder. The distribution from which the matching is sampled

depends on the algorithm used by the PRINCIPAL. In order to provide advice to an agent, the PRINCIPAL must find the probability that a maximum matching chosen from this distribution includes that agent. This problem is closely related to a computationally intractable (technically, #**P**-hard) problem, namely counting the number of maximum matchings in bipartite graphs (or sampling them uniformly) [18, 28].

One way to estimate this probability is as follows: given an algorithm that generates a random maximum matching, sample a large number of maximum matchings and compute the ratio of the number of matchings in which x^* was matched to the total number of samples.

The advice framework. The following are the steps in the maximum matching advice framework, given the set of agents $\mathbb{X}$ and the set of resources $\mathbb{Y}$.

1. An agent x^* approaches the PRINCIPAL seeking advice. The inputs to the framework are a compatibility graph G, the restrictions set R of x^*, and its incompatibility set Γ.
2. The PRINCIPAL computes (or estimates) the probability that x^* is matched to a resource and provides this information to x^*.
3. If x^* is not satisfied with the probability, then, it specifies the cost $\rho(\cdot)$ of relaxing its restrictions and a budget β as an upper bound for the cost it is willing to pay.
4. The PRINCIPAL suggests a relaxation solution (if one exists) that results in an *augmented compatibility graph* G' for which the improvement in probability of the agent being matched is maximized under the budget constraint.

The Probability Gain. Let us denote G as the original compatibility graph and G' as the new compatibility graph obtained by adding edges after relaxing the restrictions R^* chosen by the special agent x^*. Denote by $p(G)$ and $p(G')$ the probability that x^* is matched in a maximum matching of G and G' respectively. The probability gain $g(R^*)$ is defined as $p(G')-p(G)$. Since $p(G)$ does not change when x^* relaxes some restrictions, maximizing $g(R^*)$ is equivalent to maximizing $p(G')$. Now, we define the MATCHINGADVICE problem formally.

Problem MATCHINGADVICE.

Given: A bipartite compatibility graph $G(\mathbb{X}, \mathbb{Y}, E)$, an agent $x^* \in \mathbb{X}$ seeking advice, its set of restrictions R, the cost of removing each restriction, incompatibility set Γ, and a budget β.
Requirement: A set of restrictions R^* with $\rho(R^*) \leq \beta$ such that removal of R^* maximizes the gain in probability $g(R^*)$.

2.2 An Example

We use the following example of matching courses to classrooms (see Figure 1). Each classroom is a resource and each course (or instructor) is an agent. Each classroom has two attributes: capacity and region where it is located. Each course has restrictions such as the required minimum capacity and desired regions.

Agent x^* prefers a classroom of size at least 40 and regions in the order $r_1 > r_2 > r_3$. In the example of Figure 1, x^* is incompatible with all resources to begin with. To model the capacity restrictions, we discretize the relaxation: we will assume that x^* relaxes the capacity constraint in steps of 10. Accordingly, we have labels c^i_{10}, where c denotes the capacity and i denotes the step. For example, the capacity labels associated with edge $\{x^*, y_1\}$ are c^1_{10} and c^2_{10} as x^* must relax its capacity constraint by 20 for it to be compatible with y_1 with respect to capacity. There is an option to increase the capacity by adding more seats (for a fee). Again, we assume that the seating capacity can be increased in steps of 10. This is represented by labels s^i_{10}. Relaxing capacity constraint by 10 is same as increasing seating capacity by 10. Hence, as seen in Figure 1, there are three ways for y_1 to become compatible with x^*: reduce capacity requirement by 20 (remove labels c^1_{10} and c^2_{10}), increase seating capacity by 20 (remove labels s^1_{10} and s^2_{10}), or reduce capacity requirement by 10 and increase seating capacity by 10 (remove labels c^1_{10} and s^1_{10}). For the region constraint, we have one label r_i for every region i. For y_3 to be compatible with x^*, both r_1 and r_2 must be removed. This is equivalent to saying that the restriction that the classroom be located in regions 1 or 2 is relaxed.

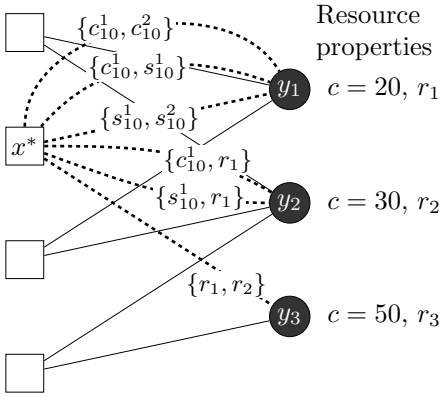

Fig. 1. A course-classroom example of matching advice framework. The agent x^* requires that the classroom capacity be at least 40 and located in region 1.

2.3 Incompatibility types

In this work, we consider advice frameworks with different forms of incompatibility relationships.

Single-Choice-Multi-Restriction incompatibility. In an incompatibility set with this property, there is exactly one choice for relaxing restrictions for each incompatible resource. This means that in the incompatibility set Γ, for every incompatible resource $y \in \mathbb{Y}_I$, there exists exactly one resource–restrictions

pair (y, R'). Note however that $|R'|$ may be ≥ 1; i.e., more than one restriction may need to be removed to make y compatible with the agent.

Multi-Choice-Single-Restriction incompatibility. In an incompatibility set with this property, for each resource–restriction pair $(y, R') \in \Gamma$, $|R'| = 1$. This means that only one restriction needs to be removed in order to make any resource compatible. However, it is possible that there are multiple choices of restrictions to remove.

We also consider **Single-Choice-Single-Restriction** incompatibility where for a resource, there is exactly one choice of one restriction to be removed to make it compatible. Similarly, we have **Multi-Choice-Multi-Restriction** incompatibility, a special case of which is the threshold-like incompatibility described below.

Threshold-like incompatibility. This type of incompatibility is motivated by capacity and region restrictions (as in Example 1). In this case, the restrictions set R can be partitioned into α blocks or *attributes* R_ℓ, $\ell = 1, 2, \ldots, \alpha$. In each $R_\ell = \{r_{\ell,1}, r_{\ell,2}, \ldots, r_{\ell,t(\ell)}\}$, the restrictions can be ordered $r_{\ell,1} < r_{\ell,2} < \ldots < r_{\ell,t(\ell)}$, where $t(\ell) = |R_\ell|$. The incompatibility set Γ satisfies the following property: $\forall (y, R') \in \Gamma$, if $r_{\ell,s} \in R'$, then, it implies that $r_{\ell,s+1} \in R'$ (if $r_{\ell,s+1}$ exists). In other words, if a restrictions set R' includes $r_{l,s}$, then it also includes all higher elements $r_{l,s+1}, r_{l,s+2}$, ... (provided they exist). Let $r_{\ell,s}$ be the minimum element in $R_\ell \cap R'$. It can be considered as the *threshold* corresponding to the ℓth attribute induced by the agent's restrictions. If a resource is incompatible with regard to the ℓth attribute, it means that the value of the resource with respect to that attribute is less than $r_{l,s}$. In the above example, the threshold for capacity is 40. Any classroom with capacity less than 30 is below the threshold and hence is incompatible.

We also use abbreviated forms when necessary. For example, the short form for Single-Choice-Multi-Restriction is Single-C-Multi-R.

3 Preliminaries

Here, we present some preliminary results regarding maximum matching size and matching probability computation.

Lemma 1. *Let G denote the original compatibility graph and $G' \neq G$ denote the compatibility graph obtained after some restrictions of agent x^* are removed.*

1. *Any maximum matching in G' that is not a maximum matching in G matches agent x^*. In addition, the edge from x^* to the matched resource is not in G.*
2. *The size of a maximum matching in G' is at most one more than that of G.*

Proof (Idea): We use the simple fact that each new edge added to G' is incident on x^*. For details, see Section A of the appendix. □

Definition 1. ***Scenarios:*** *Let G and G' denote respectively the original compatibility graph and the one that results after some restrictions of agent x^* are removed. There are two possible scenarios depending on the sizes of maximum matchings of G and G'.*

1. **Scenario 1.** *Maximum matching size in G' is one more than that of G. In this case, x^* is matched in all maximum matchings in G'. Thus, in this scenario, the probability that x^* is matched has the maximum possible value of 1.*
2. **Scenario 2.** *Maximum matching size in G' is the same as that of G. In this case, all maximum matchings of G' which are not maximum matchings in G will have x^* matched to a resource.*

4 Hardness results

In this section, we present computational intractability results for MATCHING-ADVICE. To do this, we first define the decision version of MATCHINGADVICE, which we denote by D-MATADV, as follows.

Decision Version of MATCHINGADVICE (D-MATADV) :
Given: A compatibility graph $G(\mathbb{X}, \mathbb{Y}, E)$, a special agent $x^* \in \mathbb{X}$ seeking advice, its set of restrictions R, the cost of removing each restriction, the incompatibility set Γ, a budget β, and a required benefit ψ.
Question: Is there a set of restrictions R^* with $\rho(R^*) \leq \beta$ such that the gain in probability $\mathrm{g}(R^*)$ is at least ψ?

The following result establishes the complexity of D-MATADV for the Multi-C-Single-R advice framework.

Theorem 1. D-MATADV *is **NP**-hard for the Multi-C-Single-R advice framework.*

Proof (Idea): Our reduction is from the MAX-COVERAGE problem which is known to be **NP**-complete [11]. For details, see Section B of the appendix. □

Since the Multi-C-Single-R incompatibility is a special case of threshold-like incompatibility, the following holds.

Corollary 1. D-MATADV *is **NP**-hard for the threshold-like advice framework.*

5 Algorithms for Advice Frameworks

5.1 Notation

Let G denote the compatibility graph before any restriction of the special agent x^* is removed. For a subset of restrictions $A \subseteq R$, let G_A denote the compatibility graph obtained by removing/relaxing A and let $f(A)$ denote the number of new maximum matchings in G_A. By Part (1) of Lemma 1, x^* is matched in all these matchings. We call $f(\cdot)$ the *new matchings count* function. Using this notation, G corresponds to $G_\varnothing$ and $f(\varnothing)$ equals to 0, where $\varnothing$ is the empty set. Note that the probability that x^* is matched in G, $p(G)$ (defined in Section 2) increases with $f(\cdot)$. We will use the standard definitions of monotone, submodular and supermodular functions [1]. (For the reader's convenience, these definitions are included in Section C of the appendix.) For simplicity, we use "monotone" to mean "monotone non-decreasing".

5.2 Scenario Identification

We recall from Lemma 1 and Definition 1 that two scenarios are possible when edges incident with x^* (meeting budget constraint) are added to the compatibility graph. Further, in the case of Scenario 1, the probability of matching x^* is 1; therefore, the probability that x^* is matched needs to be estimated only for Scenario 2. In this section, we will show an efficient method to (i) determine whether Scenario 1 exists, and if so, (ii) find the set of restrictions to relax. If the situation corresponds to Scenario 2, the algorithm returns the new incompatibility set Γ' of x^*.

Our method crucially uses the Dulmage-Mendelsohn (DM) decomposition of the node set of G [6,25]. Under this decomposition, any maximum matching M in a bipartite graph $G(\mathbb{X}, \mathbb{Y}, E)$ defines a partition of $\mathbb{X} \cup \mathbb{Y}$ into three sets: odd ($\mathcal{O}$), even ($\mathcal{E}$) and unreachable ($\mathcal{U}$). A node $u \in \mathcal{E}$ (respectively, $\mathcal{O}$) if there is an even (odd) length *alternating path*[4] in G from an unmatched node to u. A node $u \in \mathcal{U}$, that is, it is unreachable, if there is no alternating path in G from an unmatched node to u. We will use the following well-known results.

Lemma 2 (Irving et al. [16]). *Consider a bipartite graph $G(\mathbb{X}, \mathbb{Y}, E)$ and let $\mathcal{E}$, $\mathcal{O}$ and $\mathcal{U}$ be defined as above with respect to a maximum matching M of G.*

1. *The sets $\mathcal{E}$, $\mathcal{O}$ and $\mathcal{U}$ form a partition of $\mathbb{X} \cup \mathbb{Y}$, and this partition is independent of the maximum matching.*
2. *In any maximum matching M of G the following hold.*
   (a) *M contains only $\mathcal{UU}$ and $\mathcal{OE}$ edges.*
   (b) *Every vertex in $\mathcal{O}$ and every vertex in $\mathcal{U}$ is matched by M.*
   (c) *$|M| = |\mathcal{O}| + |\mathcal{U}|/2$.*
3. *There is no $\mathcal{EU}$ edge or $\mathcal{EE}$ edge in G.*

Lemma 3. *Let M be a maximum matching and $x \in \mathbb{X} \cap \mathcal{E}$. Adding edge $\{x, y\}$, where $y \in \mathbb{Y}$ is an incompatible resource, increases the matching size iff $y \in \mathcal{E}$.*

A proof of this lemma appears in Section D of the appendix. The method to identify Scenario 1 is described in Algorithm 1.

Correctness of Algorithm 1. We will now show that the algorithm detects Matching Scenario 1, if it exists. We note that this scenario can happen if and only if the following two conditions are met: (i) there exists a resource $y \in \mathcal{E}$ and (ii) there exists a resource-restrictions pair (y, R') such that $\rho(R') \leq \beta$. The first condition is due to Lemma 3, while the second follows from the budget constraint. The algorithm checks for precisely these conditions. Hence, it detects Scenario 1 if it exists. Also, note that the algorithm filters out resource-restrictions pairs that do not meet the budget constraint.

Lemma 4. *Algorithm 1 runs in time $O(m\sqrt{n} + |\Gamma|))$, where n and m are the number of nodes and edges in G and Γ is the incompatibility set of x^*.*

[4] Given a matching M, an alternating path between two nodes is a path in which edges in M and edges not in M alternate [25]. The length of such a path is the number of edges in the path.

Algorithm 1: Detecting Matching Scenario 1 and updating the incompatibility set

Input : Agents $\mathbb{X}$, Resources $\mathbb{Y}$, compatibility graph G, special agent x^*, its incompatibility set Γ and budget β.
Output: Decide if Matching Scenario 1 has occurred or not. If not, output the new incompatibility set Γ' that accounts for the budget.

Set $\Gamma' = \emptyset$
Compute the DM-decomposition of $\mathbb{X} \cup \mathbb{Y}$ into $\mathcal{O}$, $\mathcal{U}$, and $\mathcal{E}$
for *each* $(y, R') \in \Gamma$ **do**
  if $\rho(R') \leq \beta$ **then**
    if $y \in \mathcal{E}$ **then**
      return "Matching Scenario 1 detected" and R'
    else
      $\Gamma' \leftarrow \Gamma' \cup \{(y, R')\}$.
return "Matching Scenario 2 detected" and Γ'

Algorithm 2: Greedy algorithm for Multi-C-Single-R corresponding to Matching Scenario 2 with uniform probability of choosing a maximum matching and uniform cost for relaxing restrictions

Input : Agents $\mathbb{X}$, Resources $\mathbb{Y}$, compatibility graph G, special agent x^*, its incompatibility set Γ, budget β and an oracle for the probability $p(G)$ that x^* is matched in the compatibility graph G.
Output: Set of restrictions $A^* \subseteq R$, $|A^*| \leq \beta$

$A^* = \varnothing$.
while $|A^*| < \beta$ **do**
  $r^* = \arg\max_{r \in R} p(G_{A^* \cup \{r\}})$.
  $A^* \leftarrow A^* \cup \{r^*\}$ and $R \leftarrow R \setminus \{r^*\}$.
return A^*

Proof: To compute the DM-decomposition, we need to first compute a maximum matching M. This takes $O(m\sqrt{n})$ time using the Hopcroft-Karp algorithm [4]. Given M, computing the DM-decomposition can be done in $O(m)$ time [16, 25]. Using this decomposition, checking whether a node y is in $\mathcal{E}$ can be done in $O(1)$ time. Since for each $(y, R') \in \Gamma$, the value $\rho(R')$ can be precomputed, checking whether $\rho(R') \leq \beta$ can also be done in $O(1)$ time. Thus, each iteration of the for loop in Line 3 uses $O(1)$ time. Hence, the total time used by the loop is $O(|\Gamma|)$. Therefore, the running time of the algorithm is $O(m\sqrt{n} + |\Gamma|))$. ■

5.3 Multi-Choice-Single-Restriction

Here, we consider the Multi-Choice-Single-Restriction incompatibility framework where any resource can be made compatible with the removal of exactly one restriction. We will assume throughout that the cost of removing any restriction

is 1 and that the maximum matching algorithm samples matchings uniformly from the space of all maximum matchings. We note that for the latter case, the probability of x^* being matched is the fraction of the maximum matchings of the given compatibility graph in which x^* is matched.

Lemma 5. *Consider the Multi-C-Single-R incompatibility. Then, for Matching Scenario 2, the new matching count function $f(\cdot)$ is monotone submodular.*

For a proof of the above lemma, see Section E of the appendix. Since f is monotone submodular, we can use the greedy algorithm that iteratively picks a restriction with the highest benefit-to-cost ratio to relax [20]. Since each addition has the same cost (namely, 1), the highest benefit-to-cost ratio is achieved by a restriction that has the highest benefit. The resulting algorithm, which provides an approximation for the Multi-C-Single-R case, is shown as Algorithm 2. Note again that in the algorithm, we are using the fact that $p(\cdot)$ increases with $f(\cdot)$. The following result is again due to the fact that f is a monotone submodular function; see Section E of the appendix for a proof of the following result.

Theorem 2. *Consider the Multi-C-Single-R incompatibility. Suppose each restriction has the same removal cost and the maximum matchings of the compatibility graph G are chosen from the uniform distribution. Then, given an oracle for computing the probability $p(\cdot)$, Algorithm 2 provides a solution to the* MatchingAdvice *problem with cost at most β and benefit at least $(1 - 1/e)$ of the optimal solution.*

Suppose the incompatibility set satisfies single restriction and single choice properties. Then, it can be shown that f is monotone and modular, in which case, the greedy algorithm is optimal [8].

Corollary 2. *Consider the* MatchingAdvice *problem under single restriction and single choice incompatibility. Suppose each restriction has the same removal cost and the maximum matchings of the compatibility graph G are chosen from the uniform distribution. Then, Algorithm 2 is optimal.*

Proof. The proof is similar to that of Theorem 2. In the proof of Lemma 5, we note that $|B_y| = 1$ since the incompatibility set satisfies the single choice property. Therefore, the set C_4 is empty or $\omega_4 = 0$. Hence, $f(B) + f(B') = f(B \cup B') + f(B \cap B')$, and therefore, f is modular. Hence, the greedy strategy in Algorithm 2 gives an optimal solution [8]. ■

5.4 Threshold-like Incompatibility

We now describe an algorithm for finding an optimal solution to the MatchingAdvice problem for threshold-like incompatibility. We assume that the budget β and cost of removing each restriction are non-negative integers. Let $R = \biguplus_{1 \le \ell \le \alpha} R_\ell$ be a partition of variables where each part contains variables corresponding to values of an attribute. We say that an α-tuple $(\beta_1, \beta_2, \ldots, \beta_\alpha)$ of non-negative integers is an α-partition of the budget β if $\sum_{\ell=1}^{\alpha} \beta_\ell = \beta$. Let Π_β^α

denote all the α-partitions of β. Algorithm 3 exhaustively explores all possible budget allocations to the attributes. Once the budget is allocated, the best solution among the restrictions in each R_ℓ can be computed using a binary search. We identify the least restriction r^*_ℓ in R_ℓ such that the sum of costs of all $r \geq r^*_\ell$ in R_ℓ are removed. Unlike the previous cases, this algorithm does not assume uniform cost or uniform probability of picking a maximum matching.

Algorithm 3: Algorithm for threshold-like incompatibility corresponding to Matching Scenario 2

Input : Resources $\mathbb{Y}$, agents $\mathbb{X}$, compatibility graph G, special agent x^*, its incompatibility set Γ, budget β and a probability oracle $p(\cdot)$.
Output: Set of restrictions $A \subseteq R$ with $\rho(A) \leq \beta$

$p^* = p(G)$ and $A^* = \varnothing$
for *each* $(\beta_1, \beta_2, \ldots, \beta_\alpha) \in \Pi^\alpha_\beta$ **do**
  for $\ell = 1, 2, \ldots, \alpha$ **do**
    $r^*_\ell = \arg\min_{r \in R_\ell} \sum_{r' \geq r} \rho(r') \leq \beta_\ell$.
    Let $A_\ell = \{r \mid r \in R_\ell,\, r \geq r^*_\ell\}$.
  Let $A = \bigcup_\ell A_\ell$.
  if $p(G_A) > p^*$ **then** $A^* = A$, $p^* = p(G_A)$
return A^*

Theorem 3. *For* MatchingAdvice *with threshold-like incompatibility where the budget β and the cost of removing each restriction are non-negative integers, given an oracle for probability $p(\cdot)$, Algorithm 3 provides an optimal solution in $O(\beta^\alpha \log |R|)$ calls to the probability $p(\cdot)$ computing oracle, where R is the restrictions set of special agent x^*, and α is the number of blocks in $\mathbb{Y}$.*

Our proof of the above result appears in Section F of the appendix.

6 Computing Matching Probability

A crucial component of the advice framework is to estimate the probability that a maximum matching chosen uniformly randomly from the set of all maximum matchings includes x^*. This problem is closely related to a computationally intractable (technically, #**P**-hard) problem, namely counting the number of maximum matchings in bipartite graphs [18, 28]. In our case, this probability computation must be repeatedly performed each time a possible solution is to be evaluated. Our goal here is to reduce the number of such computations. We will show that under certain independent sampling of matchings, one can precompute a relatively small number of probabilities that can be used to find the probability of x^* being matched after relaxing any set of restrictions.

Suppose the set $\mathbb{Y}$ of resources can be partitioned into η blocks $\mathbb{Y} = \mathbb{Y}_1 \uplus \mathbb{Y}_2 \uplus \cdots \uplus \mathbb{Y}_\eta$ such that for any set of restrictions $A \subseteq R$ and any block $\mathbb{Y}_\ell$, relaxing A either makes all resources in $\mathbb{Y}_\ell$ compatible or none of its resources compatible

Fig. 2. The results for Multi-C-Single-R and Single-C-Multi-R on random bipartite graphs for varying number of restrictions and budget. The range of values on the y-axis of all plots are the same.

with x^*. Let G_A denote the compatibility graph after the restrictions in A are removed. Under the assumption that the matchings are sampled independently of one another from G_A, to compute the probability after relaxing A, it is enough to know the probability value p_ℓ that x^* is matched to a resource in $\mathbb{Y}_\ell$, $1 \leq \ell \leq \eta$ when sampled from all possible maximum matchings. Let p_0 denote the probability that x^* is not matched. Let $\mathcal{R}(A)$ denote the set of blocks whose resources become compatible with x^* after relaxing A. Then, probability that x^* appears in a maximum matching after relaxing A is given by $\frac{\sum_{\mathbb{Y}_\ell \in \mathcal{R}(A)} p_\ell}{p_0 + \sum_{\mathbb{Y}_\ell \in \mathcal{R}(A)} p_\ell}$. The justification for the summation used here is that every maximum matching containing x^* has exactly one resource matched to it. Therefore, the events that x^* is matched to a resource in $\mathbb{Y}_\ell$, $1 \leq \ell \leq \eta$ are disjoint.

We note that the number of resources m is a trivial upper bound for η, the number of blocks. In the case of threshold-like incompatibility, another upper bound can be specified. For budget β and number of attributes α, the number of optimal solutions is bounded by β^α (Algorithm 3). This serves as an upper bound for η.

7 Experimental Results

We experimented extensively on real-world and synthetic datasets to evaluate our algorithms for the advice frameworks considered.

Datasets. We considered a family of synthetic graphs and real-world datasets. We used Erdös-Renyi random bipartite graphs [7] $G(n, p)$ for experiments to evaluate the greedy algorithms for the Multi-C-Single-R and Single-C-Multi-R incompatibility frameworks. For the threshold-like incompatibility, we considered two real-world datasets. The first dataset is the *Course-Classroom* (CoCl) dataset. This comes from a university[5] for the year 2018–2019[6]. In the experiments we focused on a two-hour slot on a specific day of the week (Tuesday), and used all the courses that are scheduled in this time slot and all available rooms. There are 144 classrooms and 154 courses. Each classroom has four attributes:

[5] Bar-Ilan University, Ramat Gan, Israel.
[6] Dataset is available at https://github.com/yohayt/RAR_EUMAS2022

its *capacity*, the *region* to which it belongs, whether it allows students with physical disability and whether it allows students with hearing disability. Following the COVID-19 epidemic, additional features were added to the classes such as whether the class has facility for remote learning (https://zoom.us/ in this case). If the classroom has no feature for remote learning, then the teacher must bring the required equipment. Another feature was flexibility to add chairs to a class to increase its capacity. So, we have the attribute-augmented dataset CoCl-zc with the following extra features compared to CoCl: (i) adding chairs as an alternative to reducing capacity, (ii) remote learning in the classroom, and (iii) portable Zoom equipment as an alternative to (ii). Note that CoCl corresponds to Single-C-Multi-R threshold-like incompatibility, while CoCl-zc corresponds to Multi-C-Multi-R threshold-like incompatibility. Even though assigning classrooms to courses is well-studied [24] we did not find any publicly available dataset. The *Children Summer Vacation Activities* or *Passeport Vacances* (PassVac) [29] corresponds to online registration for assigning holiday activities to children. There are three attributes – minimum and maximum permissible age for participation with ranges. In addition, each child has restrictions as to which activity they would like to participate in. The minimum and maximum age restrictions each correspond to a threshold function. Note that the activity might be either too trivial for the child if the minimum age is relaxed, or the child may not fully understand the activity if the maximum age is relaxed. The numbers of children and activities are 634 and 533, respectively. We focused on one of the vacations in the dataset. In this vacation, there were 249 activities.

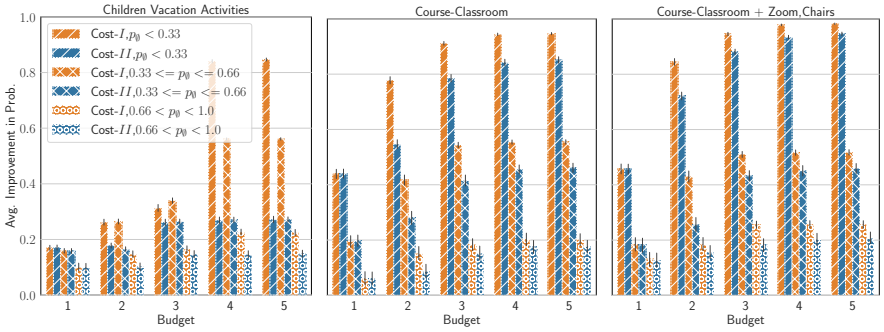

Fig. 3. The benefit obtained by removing restrictions for the threshold-like incompatibility on (i) PassVac, (ii) CoCl, and (iii) CoCl-zc. For analysis, we have partitioned the agents based on their original estimated probability of matching $p_\varnothing$.

Probability computation. In all the experiments, the probability that the special agent x^* is matched was estimated in the following manner. A random maximum matching was generated by first randomly permuting the set of agents and using the resulting compatibility graph as input to the Hopcroft-Karp algorithm [15]. Each time, 1000 such maximum matchings were generated. The

probability of x^* being matched is simply the ratio of total number of matchings in which x^* is matched to 1000.

Multi-C-Single-R with synthetic graphs. We generated 100 random bipartite graphs, each with 40 agents and 20 resources. Each edge has the probability of 0.2 to be matched. Then an additional agent was generated as x^*. A subset of restrictions was generated randomly for each resource. If the generated set is empty, then that resource would be made compatible with the agent. We experimented extensively on synthetic graphs to evaluate the greedy approach of Algorithm 2 by varying the size of the restrictions set and budget size. We used exhaustive search to obtain a pseudo-optimal solution (since the probabilities are only estimates) and compared it with the greedy solution. For each instance, we ran 100 experiments of finding sets of restrictions to be removed. The results are in the first two parts of Figure 2. We varied the maximum number of restrictions allotted per resource from 2 to 4. In the top left plot, we fixed the budget β to 2 and varied the number of restrictions from 5 to 11. In the bottom left plot, we fixed the number of restrictions to 19 and varied the budget β from 1 to 5. We observe that in each case, the greedy algorithm closely matched the performance of the pseudo-optimal solution. We note the gain is high for small budgets as only one restriction per resource needs to be removed to make it compatible. Therefore, increasing the budget only increases the gain marginally. Increasing the number of restrictions does not have much effect on the benefit.

Single-C-Multi-R incompatibility with synthetic graphs. Here, we apply the greedy algorithm (Algorithm 2). It is known to have performance guarantee of γ times the best solution given the budget, where γ is the submodularity ratio [2]. Again, we used exhaustive search to obtain a pseudo-optimal solution and compared it with the greedy solution. The experiment design is similar to that of the Multi-C-Single-R case. In the third and fourth parts of Figure 2, the results are presented for varying sizes of restrictions and budget. The number of restrictions per resource is at most 4. We note that unlike the Multi-C-Single-R case, the probability of being matched decreases with increase in the number of restrictions; this is because all the restrictions corresponding to a resource must be removed for it to become compatible. Also, increasing the budget provides significant benefit in this case as many more restrictions must be removed for resources to become compatible compared to the Multi-C-Single-R case.

Threshold-like incompatibility with real data sets. For CoCl dataset, we used 140 courses out of the 154 available and for each agent, each cost function, and each budget value, we have 100 replicates. For PassVac, we used 603 children out of the 634 children. For each agent, each cost function, and each budget value, we have 30 replicates. For both PassVac and CoCl datasets, we used two cost schemes: Cost-I is the uniform cost function where all attribute values have cost 1 and Cost-II is a linear cost function, where, for a given attribute, the cost of removing the first restriction is 1, the second is 2, and so on. Therefore, if t labels corresponding to an attribute are removed, the cost incurred is $t\,(t+1)/2$ (the more the Principal deviates from the threshold set by the agent, the higher the regret or penalty). Here, we considered multiple

agents, one at a time in our analysis. These agents were categorized based on their initial probability of being matched, $p_{\varnothing}$: (i) $[0, 1/3)$, (ii) $[1/3, 2/3)$, and (iii) $[2/3, 1)$. The results shown in Figure 3 are discussed below.

Application specific observations. For CoCl, hearing disability feature is typically the first to be relaxed. This seems to suggest that for the number of students with hearing disability, the number of classrooms which can accommodate their needs is not adequate. In the PassVac case, the abrupt increase in probability was due to a large number of activities being ranked as low preference by multiple agents (children). These are a few observations that can help the Principal to better cater to the needs of the agents.

Single-C-Multi-R vs. Multi-C-Multi-R in the CoCl dataset. We recall that the CoCl dataset has two scenarios: with Zoom and chairs and without these facilities. We note that there is not much difference in the benefits. In particular, we can see that there is no significant decrease in the improvement compared to the case when not having these facilities. This seems to indicate that the university is well prepared to the COVID-19 special needs.

Increase in benefit with budget. We observe that as the budget increases, in the case of CoCl, the probability of being matched increases gradually under both cost schemes. Also, the benefits for both cost schemes are comparable. However, in the case of PassVac, we observe an interesting threshold effect in the case of Cost-I. For example, when $p_{\varnothing} < 0.33$, until a budget of 4, there is no appreciable increase in the probability. However, for $\beta = 4$, the probability is almost 1. We observed a similar phenomenon in the case of Cost-II for budget 6, which is not presented in the plot. This knowledge of the required budget can help us to give agents an indication of the budget needed to achieve a reasonable improvement in probability. In addition, further analysis of the activities that become available after the threshold is exceeded can give us an indication of whether we can modify some of these activities to make them available with a lower budget.

8 Limitations and Future Work

A natural direction for future work is to extend the framework to allow changes to the restrictions of multiple agents. In such cases, an optimal allocation solution (e.g., Nash equilibrium [22]) can be considered. Our work assumes that each agent is matched to a single resource. So, another direction is to extend the advice framework by allowing agents to specify the number of resources needed. In such a case, when an agent does not receive the requested number of resources, the agent may be advised to either change her restrictions or reduce the number of requested resources. We note that our framework can be extended to many scenarios where resources can be shared. In such cases, for a shared resource, one can simply create copies of resources with identical properties.

Acknowledgments: We are grateful to the reviewers of EUMAS 2022 for carefully reading the manuscript and providing valuable suggestions. This work was supported by Israel Science Foundation under grant 1958/20, the EU Project TAILOR under grant 952215, Agricultural AI for Transforming Workforce and Decision Support (AgAID) grant no. 2021-67021-35344 from the USDA National

Institute of Food and Agriculture, and the US National Science Foundation grant OAC-1916805 (CINES).

References

1. Bach, F., et al.: Learning with submodular functions: A convex optimization perspective. Foundations and Trends® in Machine Learning **6**(2-3), 145–373 (2013), http://dx.doi.org/10.1561/2200000039
2. Bian, A.A., Buhmann, J.M., Krause, A., Tschiatschek, S.: Guarantees for greedy maximization of non-submodular functions with applications. In: Proc. 34th ICML, Volume 70. pp. 498–507. PMLR, Online Publisher (2017)
3. Chevaleyre, Y., Dunne, P.E., Endriss, U., Lang, J., Lemaître, M., Maudet, N., Padget, J.A., Phelps, S., Rodríguez-Aguilar, J.A., Sousa, P.: Issues in multiagent resource allocation. Informatica (Slovenia) **30**(1), 3–31 (2006)
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press and McGraw-Hill, Cambridge, MA (2009)
5. Dolgov, D.A., Durfee, E.H.: Resource allocation among agents with MDP-induced preferences. J. Artif. Intell. Res. **27**, 505–549 (2006)
6. Dulmage, A.L., Mendelsohn, N.S.: Coverings of bipartite graphs. Canadian Journal of Mathematics **10**, 517–534 (1958)
7. Easley, D., Kleinberg, J.: Networks, Crowds and Markets: Reasoning About a Highly Connected World. Cambridge University Press, New York, NY (2010)
8. Edmonds, J.: Matroids and the greedy algorithm. Mathematical programming **1**(1), 127–136 (1971)
9. Felfernig, A., Friedrich, G., Jannach, D., Zanker, M.: Developing constraint-based recommenders. In: Recommender Systems Handbook, pp. 187–215. Springer, New York, NY (2011)
10. García-Soriano, D., Bonchi, F.: Fair-by-design matching. Data Mining and Knowledge Discovery **34**(5), 1291–1335 (2020)
11. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. W. H. Freeman and Co., San Francisco, CA (1979)
12. Ghoseiri, K., Haghani, A., Hamed, M., et al.: Real-time rideshare matching problem. Tech. rep., Mid-Atlantic Universities Transportation Center (2010)
13. Gilbert, F.: A guide to sharing farm equipment. https://projects.sare.org/wp-content/uploads/Sharing-Guide-2018-_-Web.pdf (2018)
14. Gorodetski, V.I., Karsaev, O., Konushy, V.: Multi-agent system for resource allocation and scheduling. In: Proc. CEEMAS. pp. 236–246. Springer, New York, NY (2003)
15. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Computing **2**(4), 225–231 (1973)
16. Irving, R.W., Kavitha, T., Mehlhorn, K., Michail, D., Paluch, K.E.: Rank-maximal matchings. ACM Transactions on Algorithms (TALG) **2**(4), 602–610 (2006)
17. Izenberg, D., Marwaha, S., Tepper, J.: Medical students who don't match through CaRMS: "it's like a scarlet letter". Healthy Debate: https://healthydebate.ca/2018/03/topic/medical-students-carms/ (2018)
18. Jerrum, M.: Two-dimensional Monomer-Dimer systems are computationally intractable. J. Statistical Physics **48**(1–2), 121–134 (1987)
19. Lü, L., Medo, M., Zhang, Y.C.: The role of a matchmaker in buyer-vendor interactions. The European Physical Journal B **71**(4), 565–571 (2009)

20. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions—I. Mathematical programming **14**(1), 265–294 (1978)
21. Nguyen, T.T., Roos, M., Rothe, J.: A survey of approximability and inapproximability results for social welfare optimization in multiagent resource allocation. Annals of Mathematics and Artificial Intelligence **68**(1-3), 65–90 (2013)
22. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. The MIT Press (1994)
23. Parameswaran, A.G., Venetis, P., Garcia-Molina, H.: Recommendation systems with complex constraints: A course recommendation perspective. ACM Trans. Inf. Syst. **29**(4), 20:1–20:33 (2011)
24. Phillips, A.E., Waterer, H., Ehrgott, M., Ryan, D.M.: Integer programming methods for large-scale practical classroom assignment problems. Computers & Operations Research **53**, 42–53 (2015)
25. Pulleyblank, W.R.: Matchings and extensions. Handbook of combinatorics **1**, 179–232 (1995)
26. Rakhra, M., Singh, R.: Internet based resource sharing platform development for agriculture machinery and tools in Punjab, India. In: Proc. 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO). pp. 636–642 (2020)
27. Roth, A.E.: On the allocation of residents to rural hospitals: a general property of two-sided matching markets. Econometrica: Journal of the Econometric Society **54**(2), 425–427 (1986)
28. Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM Journal on Computing **8**(3), 410–421 (1979)
29. Varone, S., Beffa, C.: Dataset on a problem of assigning activities to children, with various optimization constraints. Data in brief **25**, 104168 (2019)
30. Wapner, L.M.: GPS navigation apps and the price of anarchy. The Mathematical Gazette **104**(560), 235–240 (2020)
31. Zahedi, Z., Sengupta, S., Kambhampati, S.: 'Why not give this work to them?' Explaining AI-moderated task-allocation outcomes using negotiation trees. arXiv: 2002.01640 (2020)
32. Zanker, M., Jessenitschnig, M., Schmid, W.: Preference reasoning with soft constraints in constraint-based recommender systems. Constraints An Int. J. **15**(4), 574–595 (2010)
33. Zhou, W., Han, W.: Personalized recommendation via user preference matching. Information Processing and Management **56**(3), 955–968 (2019)

Appendix

A Additional Material for Section 3

I. Statement and Proof of Lemma 1

Statement of Lemma 1. Let G denote the original compatibility graph and $G' \neq G$ denote the compatibility graph obtained after some restrictions of agent x^* are removed.

1. Any maximum matching in G' that was not a maximum matching in G matches agent x^* to a new resource. In addition, the edge from x^* to the matched resource is not in G.
2. The size of a maximum matching in G' is at most one more than that of G.

Proof. Since G' is obtained by adding new edges between x^* and one or more resources that were previously incompatible, every new edge added to G to produce G' must be incident with x^*. It can be seen that every maximum matching in G' that is not a maximum matching in G must use one of the new edges added to G. Since each such edge is incident with x^*, the statement of Part 1 holds.

Let ℓ and ℓ' denote the sizes of maximum matchings in G and G' respectively. For the sake of contradiction, suppose $\ell' \geq \ell + 2$. Then, there is a maximum matching M' in G' that is not a maximum matching in G. By Part 1, M' must contain an edge $\{x^*, y_j\}$. Let M denote the matching obtained by deleting the edge $\{x^*, y_j\}$ from M'. Since every edge in M is also an edge in G, it implies that M is a matching in G with at least $\ell + 1$ edges, a contradiction to the assumption that the maximum matching size in G is ℓ. ■

B Additional Material for Section 4

I. Statement and Proof of Theorem 1

Statement of Theorem 1. D-MatAdv is **NP**-hard for the Multi-C-Single-R advice framework.

Proof: Our proof uses a reduction from the Max-Coverage problem which is defined as follows: given a universal set $U = \{u_1, u_2, \ldots, u_r\}$, a family $F = \{F_1, F_2, \ldots, F_m\}$, where $F_j \subseteq U$, $1 \leq j \leq m$, and integers q and t, is there a subfamily F' of F such that $|F'| \leq q$ and the union of the sets in F' has at least t elements? The Max-Coverage problem is known to be **NP**-complete [11].

Let I denote an instance of Max-Coverage specified using the parameters U, F, q and t. From I, we construct an instance I' of D-MatAdv as follows.

1. The agent set $\mathbb{X} = \{x^*, x_1, x_2, \ldots, x_r\}$ has $r+1$ agents while the resource set $\mathbb{Y} = \{y_1, y_2, \ldots, y_r\}$ has r resources. (Recall that $r = |U|$.) The set $\mathbb{Y}$ is in one-to-one correspondence with U and x^* is the agent seeking advice.

2. Initially, the compatibility graph $G(\mathbb{X}, \mathbb{Y}, E)$ has the r edges given by $\{x_i, y_i\}$, $1 \leq i \leq r$. These edges form a maximum matching of size r in G. (Thus, the number of maximum matchings in G is 1.)
3. Initially, no edge is incident on x^*; in other words, the probability that a maximum matching in G includes x^* is zero.
4. The set $R = \{R_1, R_2, \ldots, R_m\}$ associated with x^* is in one-to-one correspondence with the set collection $F = \{F_1, F_2, \ldots, F_m\}$.
5. Suppose $F_i = \{u_{i_1}, u_{i_2}, \ldots, u_{i_\ell}\}$, where $\ell = |F_i|$. Corresponding to F_i, we have the following set of resource-restriction pairs: $\{(y_{i_j}, \{R_i\}) \ : \ 1 \leq j \leq \ell\}$. Thus, removing restriction R_i adds the following set of ℓ edges $\{\{x^*, y_{i_j}\} \ : \ 1 \leq j \leq \ell\}$ to the initial compatibility graph G. Thus, we can think of removing restriction R_i as adding the edges from x^* to the nodes in $\mathbb{Y}$ corresponding to the elements of F_i. Since each of these edges has exactly one restriction, namely R_i, this corresponds to the Multi-C-Single-R case. Note that when these edges are added to G, the size of the maximum matching remains r. However, the number of maximum matchings increases to $|F_i|+1$, and $|F_i|$ of these maximum matchings include x^*. Thus, the increase in probability due to removing restriction R_i is $|F_i|/(|F_i|+1)$, $1 \leq i \leq m$.
6. The cost of removing each restriction R_i is set to 1. The budget β is set to q, the budget on the number of sets in the Max-Coverage instance. The required increase in probability for x^* is set to $t/(t+1)$, where t is the coverage requirement in the Max-Coverage instance.

This completes the construction of the instance I' of D-MatAdv. It is easy to see that the construction can be carried out in polynomial time. We now show that there is a solution to the D-MatAdv instance I' iff there is a solution to the Max-Coverage instance I.

If Part: Suppose the Max-Coverage instance I has a solution. Without loss of generality, we can assume that the solution F' is given by $F' = \{F_1, F_2, \ldots, F_q\}$ and that the sets in F' cover $t' \geq t$ elements of U. For the D-MatAdv instance, we choose the set R' of restrictions to be removed as $\{R_1, R_2, \ldots, R_q\}$. which is obtained by choosing the restrictions corresponding to the sets in F'. The cost of adding these variables is $q = \beta$. Since the sets in F' cover $t' \geq t$ elements of U, by our construction, the removal of the restrictions in R' adds t' edges between x^* and the resources. However, the maximum matching in the resulting graph remains r. Thus, in the new compatibility graph the number of maximum matchings of size r is $t' + 1$ and t' of these matchings contain x^*. Thus, the probability of x^* being matched in the new compatibility graph is at least $t'/(t'+1)$. Since $t' \geq t$, it can be seen that the increase in probability is at least $t/(t+1)$. Thus, the D-MatAdv instance I' has a solution.

Only If Part: Suppose the D-MatAdv instance I' has a solution. Without loss of generality, we may assume that the set R' of restrictions that are removed is $R' = \{R_1, R_2, \ldots, R_q\}$. Further, the increase in probability of x^* due to this solution is at least $t/(t+1)$. By our construction, it can be seen that the removal of R' must add at least t edges between x^* and the resources. Now, consider the subfamily $F' = \{F_1, F_2, \ldots, F_q\}$. The removal of each restriction R_i in R' adds

the edges corresponding to the elements in F_i, $1 \leq i \leq q$. Since the removal of all the restrictions in R' causes at least t edges to be added to x^*, it follows that the union of the sets in F' covers at least t elements of U. Thus, F' is a solution to the MAX-COVERAGE instance I, and this completes our proof of Part (a) of Theorem 1. ■

C Additional Material for Section 5.1

I. Definitions of Monotone, Submodular and Supermodular Functions:

Definition 2. *Let X be a set and let $\mathbb{N}$ denote the set of non-negative integers. Suppose $f : 2^X \rightarrow \mathbb{N}$ is a function.*

*(a) Function f is **monotone non-decreasing** if for any P and Q such that $P \subseteq Q \subseteq X$, $f(P) \leq f(Q)$.*
*(b) Function f is **submodular** if for any P and Q such that $P, Q \subseteq X$, $f(P) + f(Q) \geq f(P \cup Q) + f(P \cap Q)$.*
*(c) Function f is **supermodular** if for any P and Q such that $P, Q \subseteq X$, $f(P) + f(Q) \leq f(P \cup Q) + f(P \cap Q)$.*
*(d) Function f is **modular** if is both submodular and supermodular.*

D Additional Material for Section 5.2

I. Statement and Proof of Lemma 3:

Statement of Lemma 3: Let M be a maximum matching and $x \in \mathbb{X} \cap \mathcal{E}$. Adding edge $\{x, y\}$, where $y \in \mathbb{Y}$ is an incompatible resource, increases the matching size iff $y \in \mathcal{E}$.

Proof: We will use the following notation: For two nodes u and v participating in a path P, let uPv denote the subpath from u to v in P. We have three cases: (i) $y \in \mathcal{E}$, (ii) $y \in \mathcal{U}$, and (iii) $y \in \mathcal{O}$.

Case 1: $y \in \mathcal{E}$. We will show that the matching size increases if the edge $\{x, y\}$ is added. Let P_x and P_y denote even-length alternating paths from x to a free node f_x and from y to a free node f_y respectively. Note that if x is a free node, then P_x corresponds to a path of length 0 containing just x. The same holds for y as well. We will show that P_x and P_y are disjoint. Let v be a node common to both P_x and P_y. If $v \in X$, then, vP_yf_y is an odd-length alternating path, while vP_xf_x is an even-length alternating path contradicting the fact that $\mathcal{E}$ and $\mathcal{O}$ are disjoint (Lemma 2(1)). Similarly, if $v \in Y$, then, vP_xf_x is an odd-length alternating path, while vP_yf_y is an even-length alternating path, again a contradiction. Therefore, P_x and P_y are disjoint and therefore, the path $f_xP_xxyP_yf_y$ is an augmenting path in the new bipartite graph.

Case 2: $y \in \mathcal{U}$**.** We will show that the matching size does not increase when edge $\{x, y\}$ is added. Note that if adding this edge increases the matching size, then, there exists an augmenting path P with respect to the matching M. Since M is a maximum matching in the original graph, P has to necessarily contain the new edge. Also, since P is an augmenting path, both its end points are free nodes. Therefore, P is of the form $P = f_x P' x y P'' f_y$ where f_x and f_y are free nodes. If $y \neq f_y$, this implies that either y is reachable from free node f_y via the alternating path $yP''f_y$ in the original graph, a contradiction to the fact that $y \in \mathcal{U}$. Moreover, since $y \in \mathcal{U}$, y cannot be a free node, and therefore, $y \neq f_y$. Hence, P cannot exist.

Case 3: $y \in \mathcal{O}$**.** We will show that the matching size does not increase when edge $\{x, y\}$ is added. As in Case 2, suppose the matching size increases, then there exists an augmenting path P of the form $P = f_x P' x y P'' f_y$, where f_x and f_y are free nodes. If $y \in \mathcal{O}$, then, P' is an even-length path since $f_x P' x$ is odd-length, $\{x, y\}$ is odd-length, and $yP_y f_y$ is even-length. This contradicts the fact that any augmenting path has to be odd-length.

Therefore, an augmenting path cannot be formed when the edge $\{x, y\}$ is added with $y \in \mathcal{U} \cup \mathcal{O}$, and in turn, the matching size cannot increase. ■

E Additional Material for Section 5.3

I. Statement and Proof of Lemma 5

Statement of Lemma 5: Consider a matching advice framework with the Multi-Choice-Single-Restriction incompatibility. Then, for Matching Scenario 2, the new matching count function $f(\cdot)$ is monotone submodular.

Proof. Since for any A and A' such that $A \subseteq A'$, G_A is a subgraph of $G_{A'}$ and the maximum matching size in $G_{A'}$ is the same as that in G_A, it follows that any new matching in G_A is also a new matching in $G_{A'}$. Therefore, f is monotone. Now, we will show that for any $B, B' \subseteq R$, $f(B)+f(B') \geq f(B \cup B')+f(B \cap B')$. Suppose $y \in \mathbb{Y}_I$ is an incompatible resource and let $B_y \subseteq R$ be the set of restrictions such that for $r \in B_y$, removing r makes y compatible, or in other words, $(y, \{r\}) \in \Gamma$ iff $r \in B_y$. Resource y belongs to one of the following four classes.

$$y \in \begin{cases} C_1, \text{ if } B_y \cap (B \cap B') \neq \varnothing, \\ C_2, \text{ if } B_y \cap B \neq \varnothing \text{ and } B_y \cap B' = \varnothing, \\ C_3, \text{ if } B_y \cap B = \varnothing \text{ and } B_y \cap B' \neq \varnothing, \\ C_4, \text{ if } B_y \cap B \neq \varnothing,\ B_y \cap B' \neq \varnothing \text{ and} \\ \quad B_y \cap (B \cap B') = \varnothing\,. \end{cases} \tag{1}$$

Let ω_ℓ denote the number of new maximum matchings obtained by adding all edges $\{x^*, y\}$, $y \in C_\ell$ to G. Note that $f(B \cap B')$ corresponds to maximum matchings obtained from relaxing all resources belonging to C_1, and therefore, $f(B \cap B') = \omega_1$. The quantity $f(B)$ corresponds to maximum matchings resulting from adding resources from C_1, C_2, and C_4, and therefore, $f(B) =$

$\omega_1+\omega_2+\omega_4$. Similarly, $f(B') = \omega_1+\omega_3+\omega_4$. Finally, $f(B\cup B') = \omega_1+\omega_2+\omega_3+\omega_4$. Clearly, $f(B) + f(B') \geq f(B \cup B') + f(B \cap B')$. ■

II. Statement and Proof of Theorem 2:

Statement of Theorem 2. Consider the Multi-C-Single-R incompatibility. Suppose each restriction has the same removal cost and the maximum matchings of the compatibility graph G are chosen from the uniform distribution. Then, given an oracle for computing the probability $p(\cdot)$, Algorithm 2 provides a solution to the MatchingAdvice problem with cost at most β and benefit at least $(1 - 1/e)$ of the optimal solution.

Proof. Let G' denote the compatibility graph obtained after removing the restrictions obtained by Algorithm 2 and G_{opt} denote the compatibility graph obtained by removing restrictions from an optimal solution. Let ω^-_{old} and ω^+_{old} denote the number of maximum matchings in the old compatibility graph where x^* is not matched and matched respectively. Let $\omega^+_{G'}$ and $\omega^+_{G_{\text{opt}}}$ denote the number of new maximum matchings in G' and G_{opt} respectively. From Lemma 1, it follows that any new matching in the new compatibility graph (G' or G_{opt}) must have x^* matched. Hence, the total number of maximum matching in G' and G_{opt} are $\omega^-_{\text{old}} + \omega^+_{\text{old}} + \omega^+_{G'}$ and $\omega^-_{\text{old}} + \omega^+_{\text{old}} + \omega^+_{G_{\text{opt}}}$ respectively.

By Lemma 5, Algorithm 2 provides a solution for which the total number of new maximum matchings is at least $(1 - 1/e)$ times the total number of new maximum matchings in an optimal solution. Therefore, $\omega^+_{G'} \geq (1 - 1/e)\,\omega^+_{G_{\text{opt}}}$. Let $p(G') = \frac{\omega^+_{\text{old}}+\omega^+_{G'}}{\omega^-_{\text{old}}+\omega^+_{\text{old}}+\omega^+_{G'}}$ and $p(G_{\text{opt}}) = \frac{\omega^+_{\text{old}}+\omega^+_{G_{\text{opt}}}}{\omega^-_{\text{old}}+\omega^+_{\text{old}}+\omega^+_{G_{\text{opt}}}}$ denote the probability that x^* is matched in G' and G_{opt} respectively.

$$\begin{aligned}
\frac{p(G')}{p(G_{\text{opt}})} &= \frac{\omega^+_{\text{old}} + \omega^+_{G'}}{\omega^+_{\text{old}} + \omega^+_{G_{\text{opt}}}} \frac{\omega^-_{\text{old}} + \omega^+_{\text{old}} + \omega^+_{G_{\text{opt}}}}{\omega^-_{\text{old}} + \omega^+_{\text{old}} + \omega^+_{G'}} \geq \frac{\omega^+_{\text{old}} + \omega^+_{G'}}{\omega^+_{\text{old}} + \omega^+_{G_{\text{opt}}}} \\
&\geq \frac{\omega^+_{\text{old}} + (1-1/e)\omega^+_{G_{\text{opt}}}}{\omega^+_{\text{old}} + \omega^+_{G_{\text{opt}}}} \\
&= \frac{(\omega^+_{\text{old}} + \omega^+_{G_{\text{opt}}}) - (1/e)\omega^+_{G_{\text{opt}}}}{\omega^+_{\text{old}} + \omega^+_{G_{\text{opt}}}} = 1 - \Big(\frac{1}{e}\,\frac{\omega^+_{G_{\text{opt}}}}{\omega^+_{\text{old}} + \omega^+_{G_{\text{opt}}}}\Big) \\
&\geq \Big(1 - \frac{1}{e}\Big).
\end{aligned}$$

This completes our proof of Theorem 2. ■

F Additional Material for Section 5.4

I. Statement and Proof of Theorem 3:

Statement of Theorem 3: For MatchingAdvice with threshold-like incompatibility, given an oracle for probability $p(\cdot)$, Algorithm 3 provides an optimal

solution in $O(\beta^\alpha \log |R|)$ calls to the probability $p(\cdot)$ computing oracle, where β is the budget, R is the restrictions set of special agent x^*, and α is the number of blocks in $\mathbb{Y}$.

Proof. First, we will show that the algorithm provides an optimal solution. Let A be any set of restrictions. We will show that there exists an $A' \subseteq A$ such that $\rho(A') \leq \rho(A)$ and $\mathrm{g}(A') \geq \mathrm{g}(A)$ satisfying the following: for any block R_ℓ, $A' \cap R_\ell = \{r_{\ell,s}, r_{\ell,s+1}, \ldots, r_{\ell,t(\ell)}\}$ for some $1 \leq s \leq t(\ell)$. Let $\mathcal{R}_A = \{R' \mid (y, R') \in \Gamma$ and y becomes compatible after relaxing $A\}$. By definition, every R' is of the form $\{r_{\ell,s'}, r_{\ell,s'+1}, \ldots, r_{\ell,t(\ell)}\}$. We simply choose $A' = \bigcup_{R' \in \mathcal{R}_A} \bigcup_{\ell=1}^{\alpha} R' \cap R_\ell$, $1 \leq \ell \leq \alpha$. It is easy to see that A' is of the form described above where $r_{\ell,s} = \min_{R' \in \mathcal{R}_A} \min R'$. Since, $\bigcup_{R' \cup \mathcal{R}_A} R' \subseteq A$, it follows that $A' \subseteq A$. Therefore, any optimal solution must have the property of A'.

Now consider any two sets $A_1 \subseteq A_2$ that satisfy the property of A'. Since $\mathrm{g}(A_2) \leq \mathrm{g}(A_1)$, given a budget β_ℓ for block ℓ, the gain can be maximized by choosing the least element possible from the set R_ℓ. This is being performed in Line 4 in the algorithm. This proves the first part of the statement.

Now we will bound the time complexity of the algorithm. Firstly, we note that the number of α-partitions of β is $O(\beta^\alpha)$, and can be enumerated in as much time. Secondly, given a budget β_ℓ for clause ℓ, Line 4 can be computed in $O(\log |R_\ell|)$ time (using a binary search given the natural ordering of the variables in this set). ■

G Additional Material for Section 7

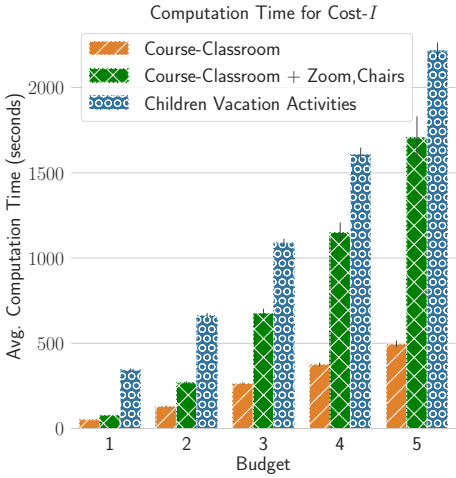

Fig. 4. Computation time for the different datasets and varying budget for Cost-I. Computation time for Cost-II was similar. The error bars represent the size of 95% confidence intervals.

Computation time of Algorithm 3: The time required to run Algorithm 3 is provided in the bar chart of Figure 4. The computation time for the PassVac dataset is much longer even for lower budget since the corresponding compatibility graph is larger compared to the CoCl dataset. Also, the addition of attributes in the CoCl dataset (Zoom and chairs) significantly increases the computation time. The same holds as the budget is increased, as there are many more partitions of the budget to evaluate.