VisMatchmaker: Cooperation of the User and the Computer in Centralized Matching Adjustment

Po-Ming Law, Wenchao Wu, Yixian Zheng, and Huamin Qu, *Member, IEEE*

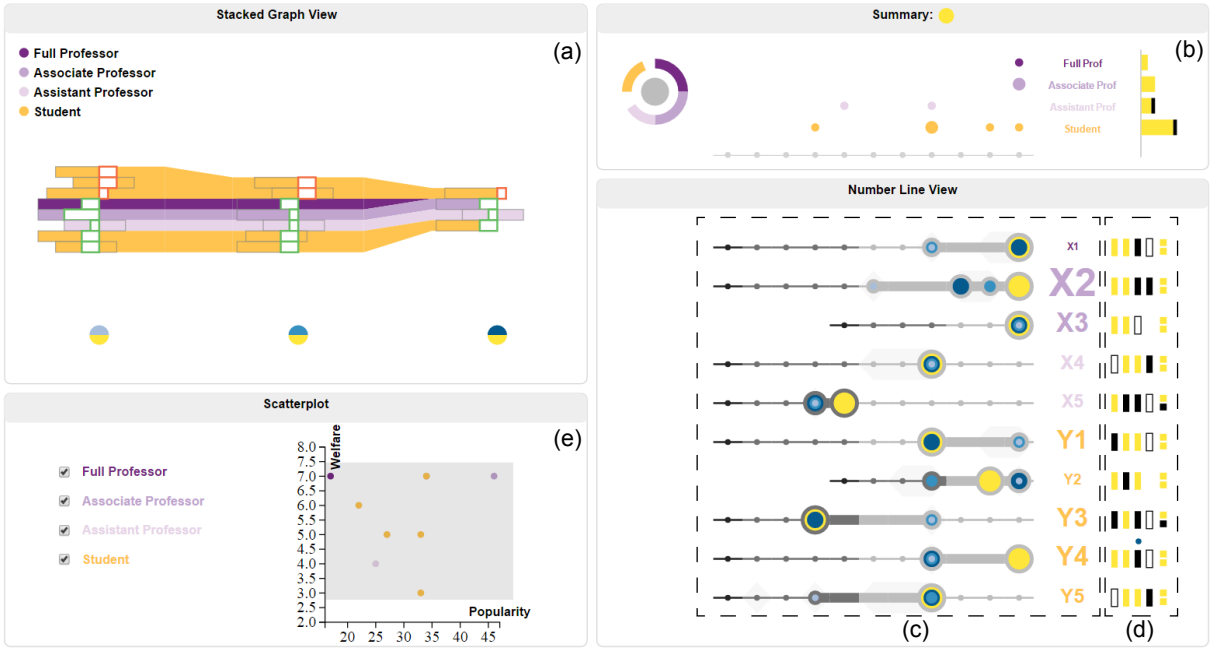

Fig. 1. An overview of the interface of VisMatchmaker. (a) The stacked graph view enables the comparison amongst multiple matchings. (b) The summary view provides different summary statistics of a selected matching. (c) The number line view shows different aspects of each agent's goodness of match in multiple matchings. (d) The visual preference list shows the states of the items in each agent's preference list. (e) The scatterplot gives an overview of the popularity and the welfare of the agents.

Abstract—Centralized matching is a ubiquitous resource allocation problem. In a centralized matching problem, each agent has a preference list ranking the other agents and a central planner is responsible for matching the agents manually or with an algorithm. While algorithms can find a matching which optimizes some performance metrics, they are used as a black box and preclude the central planner from applying his domain knowledge to find a matching which aligns better with the user tasks. Furthermore, the existing matching visualization techniques (i.e. bipartite graph and adjacency matrix) fail in helping the central planner understand the differences between matchings. In this paper, we present VisMatchmaker, a visualization system which allows the central planner to explore alternatives to an algorithm-generated matching. We identified three common tasks in the process of matching adjustment: problem detection, matching recommendation and matching evaluation. We classified matching comparison into three levels and designed visualization techniques for them, including the number line view and the stacked graph view. Two types of algorithmic support, namely direct assignment and range search, and their interactive operations are also provided to enable the user to apply his domain knowledge in matching adjustment.

Index Terms—Centralized matching, matching visualization, interaction techniques, visual analytics

1 Introduction

Centralized matching is a ubiquitous resource allocation problem. A centralized market is often created to reduce the cost of searching [14]. In a centralized market, each agent (which is a person in many cases) has a preference list ranking a subset of the other agents and a central planner is responsible for pairing the agents up. Finding a matching with desirable properties is crucial as it can affect productivity and agents' satisfaction. For example, a software company adopting pair programming may want to pair programmers who can work well together to enhance the productivity of the firm [4]; a school may want to match mentors and mentees to help students with their courses [22]; a university department may want to assign courses to professors so that the professors are satisfied with the arrangement.

The number of possible matchings is huge even with a small number of agents [14]. To find an optimal matching, a matching problem is formulated as an optimization problem which aims at optimizing one or two performance metrics [2, 11]. The most common algorithm is the Deferred-Acceptance (DA) algorithm which finds a stable matching in a two-sided matching problem [7]. While its efficiency makes it suitable for markets with a large number of agents (such as college admission), the algorithmic result may not align well with the user tasks. In a small matching problem (such as matching professors and

• *Po-Ming Law, Wenchao Wu, Yixian Zheng, and Huamin Qu are with the Hong Kong University of Science and Technology. E-mail: pmlaw@connect.ust.hk, {wwuag, yzhengaj, huamin}@cse.ust.hk.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

courses), the central planner is often knowledgeable about the situation and wants to apply his knowledge to adjust the algorithm-generated matching to make it more reasonable. However, the black-box like algorithm precludes the central planner from steering it.

When adjusting a matching, compromise in matching quality in some areas is usually needed in exchange for an improvement. To understand the trade-offs involved in different matchings and inform the central planner's decision of which matching is better, various matching approaches need to be compared. Yet, the widely used matching visualization techniques (i.e. bipartite graph and adjacency matrix) only provide information about who is matched to whom rather than the trade-offs between two different matchings. Although summary statistics such as social welfare can be used to compare matchings, they convey only the high-level idea about which matching ranks higher instead of the reasons for the rank. It can be that two matchings have the same score but the central planner prefers one of them more because of some of its desirable properties.

To address the above challenges, we propose VisMatchmaker, an interactive visualization system for adjusting an algorithm-generated matching. VisMatchmaker enables the user to issue queries of matching criteria to steer the adjustment-searching algorithm. We use a variation of Top Trading Cycle Algorithm [25] to narrow down the huge matching search space according to user's queries. The algorithm recommends to the user some matchings which are ranked based on a performance metric. The recommended matchings can then be compared for evaluation. We categorized matching comparison into three levels and created novel visualization techniques to help the user understand the trade-offs among different matchings.

The main contributions in this paper are: 1) A characterization of the problem of centralized matching adjustment. 2) An interactive visualization system for adjusting an algorithmic result with the help of an adjustment-searching algorithm. 3) A three-level classification of matching comparison and novel visualization techniques for comparing matchings at different resolutions.

2 Related Work

2.1 Matching Algorithms

A matching problem can be represented by a graph. There has been extensive research on graph matching algorithms to find maximum cardinality matching, maximum weight matching and maximum weight maximum cardinality matching [2]. When each agent has a preference list ranking the other agents, stability of matching is usually a concern. Since Gale and Shapley proposed the DA algorithm [7] to find a stable matching, algorithms are proposed to find a stable matching with other desirable properties [8]. Hass [11] summarized four common metrics (stability, welfare, fairness and number of matched pairs) for assessing the performance of a matching. Many algorithms consider matching problems a multiobjective optimization problem which optimizes a subset of these performance metrics. Kimbrough et al. [14] offered a heuristic approach to find a stable matching which is Pareto superior to those found by the DA algorithm in terms of social welfare and fairness. Nakamura et al. [20] used genetic algorithm to find a sex-fair matching in the stable marriage problem.

While these algorithms can find a matching efficiently, their black-box nature precludes the central planner from steering them and applying his domain knowledge in matching the agents. The result generated by an algorithm may not be able satisfy the user tasks and improvements are often possible if the user is enabled to bring in his domain knowledge. Hence, there exists a need to help the user explore the possible matching adjustments to better achieve his tasks.

2.2 Human Machine Collaborative Decision Making

The goal of our system is to involve people in the matching adjustment process so that the adjusted matching can align better with the user tasks. Malasky referred to the involvement of humans in the loop of problem solving to generate solutions that improve upon those generated by solely a human or solely a computer as human-machine collaborative decision-making (HMCDM) [18]. HMCDM combines the intuition of a human and the computational speed of a computer. To design such a system, the tasks to be allocated to the user and to the system need to be decided. Ad-hoc approaches determine the proper allocation of tasks using gut feeling while formal approaches assign the tasks based on the capabilities of humans and machines. A balanced approach which is a combination of both approaches is usually used in practice.

Much research has been done on a tighter coupling of the domain knowledge of the user and algorithmic support. One such area is user-centred decision tree construction. Multiple visualization systems are developed to allow the user to contribute his domain knowledge to co-create a better decision tree with algorithms [3, 15, 29].

2.3 Matching Visualization

Node-link diagram and adjacency matrix are the most popular for network visualization [19]. Node-link diagram is more intuitive and interpretable than adjacency matrix [9]. However, as the number of nodes and links increases, the links easily clutter to form a hairball. Much research is devoted to preventing cluttering or providing an easier interpretation of the hidden pattern in a large network (e.g. [21, 30]). For matching visualization, bipartite graph which is a subtype of node-link diagram is the most common for its easy interpretation. Matrix representation can also be seen in some matching games [1]. While they show who is matched to whom effectively, they fail to communicate the quality of a matching and the trade-offs among multiple matchings which is important for concluding which matching is better.

2.4 Visual Comparison

Gleicher et al. provided a thorough analysis of three categories of comparison of complex objects [10]. They concluded that juxtaposition, superposition and explicit encoding are the fundamental building blocks of comparative designs. Juxtaposition usually requires objects to be placed side by side for comparison. It relies on our memory to make connection between objects. Animation which is termed as temporal juxtaposition in their survey has been used by a lot of work (e.g. [32]). Superposition involves overlaying one object over another one which are put in the same space. Explicit encoding uses computation to find the relationship between objects thereby sparing viewers of the efforts. Interaction can also augment visual comparison by rearranging and manipulating the objects (e.g. [23]). We employed both explicit encoding and juxtaposition in the three levels of matching comparison.

2.5 Crew Scheduling

A related class of problems is called crew scheduling which can be defined as a problem of assigning a group of crews to a set of tasks [5]. The scheduling process mainly involves minimizing the cost while satisfying a set of constraints imposed by local regulations. Due to the computational complexity of finding the optimal solution, much research is devoted to developing algorithms which obtain near-optimal solutions (e.g. [6, 17]) with limited human involvement.

However, the algorithmic result often needs to be adjusted in day-to-day operation due to unpredictable events such as absenteeism [27]. Different solutions are proposed to incorporate domain knowledge into the result. Shibghatullah et al. [27] proposed a conceptual framework for developing a crew schedule management system to automate the process of crew reassignment. An interface is involved in the framework to receive the user tasks, deliver the tasks to the system and present the results to the user. Yamada et al.'s system [31] automatically proposes adjustments to the schedule and explains why the changes are proposed in words when the situation changes. The user then selects from the proposed candidates according to the explanations.

Unlike setting an anchor to fix an assignment in the scheduling stage before running an algorithm, an adjustment to an assignment in an algorithmic result will influence other assignments and cause a chain effect. Our system enables the user to adjust a matching and steer the algorithm to compute its influence to the other assignments. It is therefore more similar to the rescheduling process as both of them use a pre-allocation as an input. However, different from rescheduling which involves the schedule responding to frequent situational changes, our goal is to let the user refine a matching.

3 Background

3.1 Definition of Centralized Matching

We deal with a class of matching problem in which each agent (which is a person in many cases) has a preference list ranking a subset of the other agents. The preference list of an agent can be inferred from the attributes he is looking for. An agent's preference over the other agents must be strict meaning that an agent cannot be indifferent between two different agents. Agents can be grouped and groups can be organized hierarchically. A central planner is responsible for pairing the agents up. A special instance of centralized matching is called two-sided matching in which agents are divided into two sets and agents in one set can only be matched to the agents in another set. Our system can handle one-to-one, one-to-many and many-to-many matching problems. In a one-to-one matching problem, each agent can only be paired with at most one agent whilst in a one-to-many matching problem, some agents have quotas greater than one which allow them to be paired with more than one agent. A matching refers to the set of all pairings. We are particularly interested in small matching problems (number of agents $<$200) in which the central planner may have knowledge about the agents and want to apply his knowledge while matching the agents. This is often not true in large matching problems.

3.2 Common Performance Metrics

Hass et al. [11] summarized four common performance metrics (i.e. stability, welfare, fairness and number of matched pairs) of a matching in one-to-one matching problems. However, in small matching problems, stability may not be a concern. By examining the literature, we found that welfare, fairness and number of matched pairs are used also in one-to-many and many-to-many matching problems. They provide a summary of a matching and give a brief idea of which matching might be more desirable. Their definitions are as follows:

Welfare is a measure of satisfaction of an agent. An agent's welfare can be calculated by mapping the rank of his partner inversely to his welfare score (i.e. the lower the rank of an agent's partner, the higher his welfare score). We use the mapping adopted by Liu et al. [16]. The welfare of a group of agents is the average welfare score of the agents in the group.

Fairness is a comparison of the welfare score of two groups of agents. The closer the welfare scores of two groups, the fairer.

Number of matched pairs indicates how fully the resources are utilized. When a bipartite graph is used to represent a matching, it is the number of links in the graph.

4 Cooperation of User and Computer for Matching Adjustment

Our goal is to develop a system by which the user and the computer can co-create a matching by contributing what they do best in the process of matching adjustment: the user provides his domain knowledge to steer the adjustment-searching algorithm and evaluates the matchings recommended by the algorithm while the computer searches for the required matchings and generates visualization of them for evaluation. In this section, we present a model of matching adjustment. We first give an example to illustrate the needs for human involvement in centralized matching problems. Then, we provide a thorough analysis of the tasks involved when adjusting an algorithm-generated matching.

4.1 Motivation

While algorithms can find a matching which optimizes a subset of the performance metrics, the algorithmic result may not align well with user tasks. This provides incentive for the central planner to adjust a matching to see if improvements are possible. Indeed, some areas of a matching can often be improved by sacrificing other less important areas of the matching. Consider a university department trying to match professors and teaching assistants. Each graduate student has a preference list of professors and each professor has a preference list of graduate students. A graduate student matched to a professor who has a course to teach will be the teaching assistant of the course in that semester. Matching A in Fig. 2 is generated by the most commonly used DA algorithm which maximizes stability. Both X1 and X5 choose Y2 as their first choice but the algorithm somehow favors X5 and pairs him with Y2. Instead of enhancing matching stability, the central planner might want to exploit the knowledge of who goes well together (probably from the special requests of the professors). In the example, the central planner wants to pair X1 with Y2 because X1 has experience working with Y2. He also wants to match X2 and X3 to their first choices because of the special requests of these two professors. By matching X1, X2, X3 to their first choices and, X5 to his less desirable partners, matching B is the result. While matching B is not a stable matching, all X1, X2 and X3 get a better partner. Furthermore, the welfare of professors increases and the total welfare remains the same. The central planner might prefer B over A. Our goal is to develop a system which empowers the user to use his domain knowledge to adjust an algorithmic result.

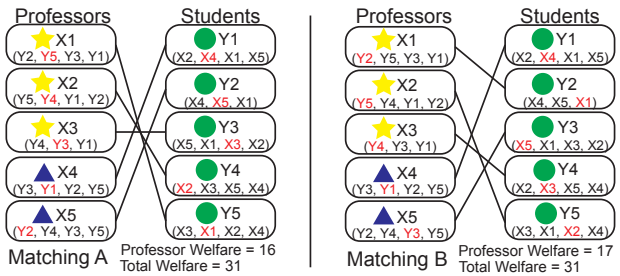

Fig. 2. Matching A is the algorithmic result. The central planner wants to match X1 to Y2 and match X2 and X3 to their first choices. Matching B is the result after the adjustment. The central planner might prefer B over A although B is not stable.

4.2 Task Analysis

To allow collaboration between the user and the computer, the task division between them needs to be decided [18]. To our best knowledge, there is no existing system which aims to involve human in matching adjustment. We collaborated with a central planner who has experience matching professors and courses as well as professors and TAs in a university department. He was also a conference chair and is knowledgeable about matching reviewers and papers. By reviewing the literature of problems with a similar nature (the literature on user involvement in decision tree construction in particular [3, 15, 29]) and interviewing our collaborator, we concluded that there are three common tasks in adjusting a matching. The central planner first needs to **detect the problems** in a matching and identify some ways to improve the matching with his domain knowledge. After the user indicates how they would like to improve the matching, an algorithm **recommends matching adjustments** based on the user's queries. The central planner then **evaluates the recommended matchings** by comparing them and assessing the trade-offs among them. The detailed requirements of the three common tasks are given in the following sections.

4.2.1 Problem Detection

After reviewing different matching problems and interviewing our collaborator, we concluded that there are two common questions the central planner wants to answer upon obtaining an algorithm-generated matching: Q1) An agent is unmatched or matched to undesirable partner(s). Is it possible to improve his match and hence his welfare? Q2) An agent is particularly popular and many agents have chosen him as their top choices. The popular agent is now paired with A. How would the result be if the popular agent is paired with agent B who also put the popular agent as his top choice? To address the two questions, the user needs to first identify the agents with undesirable matches (Q1) and the popular agents (Q2) in the algorithmic result. Hence, the welfare and the popularity of agents are required to be visualized.

Next, the user is enabled to improve the agent's match or directly assign a different agent to the popular agent with two interactive operations (i.e. direct assignment and range search). The two interactive operations allow the user to issue queries which steer the adjustment-searching algorithm to compute the influence of an adjustment to the

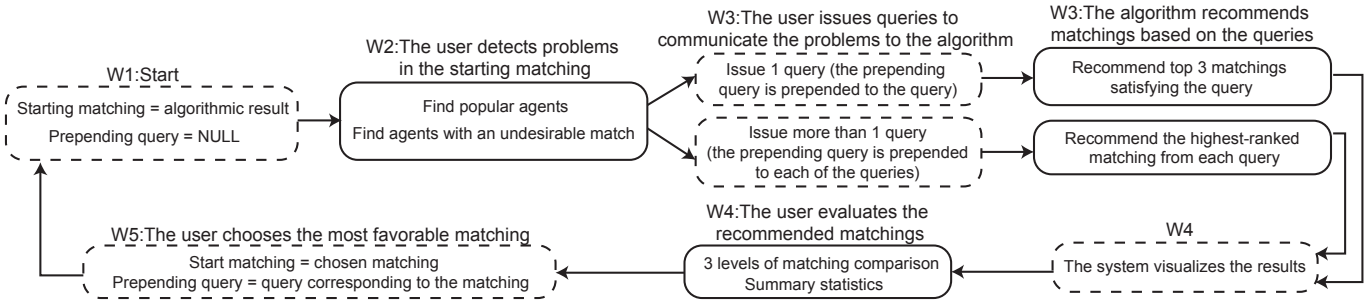

Fig. 3. The matching adjustment model. The three common tasks are represented by the boxes with a solid outline and the linkages between the tasks are represented by the boxes with dotted lines.

whole matching and recommend matchings to the user. Direct assignment instructs the algorithm to find matchings in which an agent is directly assigned to another agent while range search forces the algorithm to search for matchings which pair an agent with partner(s) within a range of goodness. For example, range search can be used to search for matchings in which an agent is paired with better partner(s) than he currently has. For flexible matching exploration, a query can be formed by combining multiple direct assignments and range searches (e.g. assign A to B AND assign C to D AND improve the match of E). After issuing a query, matching recommendations are found accordingly. Our system enables the user to compare the recommended matchings from different queries. For example, the user can compare the highest-ranked matching from the query "assign A to B" and that from another query "assign A to C".

4.2.2 Matching Recommendation

Required by our collaborator, several matching results should be recommended by the systems after the user indicates adjustments through direct assignments and range searches. The matchings are ranked in descending order of total welfare as total welfare is a reasonable measure of overall satisfaction of the agents.

We focus on a subset of matchings which are reasonable and comparable. By reasonable, the matchings recommended by the system should not have a large number of unmatched agents. For easy comparison, the matchings recommended should not be drastically different from the algorithmic result. Only a limited number of agents should have their partners changed. Besides easy comparison, limited differences between the recommended matchings and the algorithmic result make the computation of the influence of an adjustment to the whole matching less expensive. It also allows the central planner to progressively improve the algorithmic result step by step instead of drastically changing it once and for all. Progressive improvement allows the user to have a better control over the matching adjustment process.

After recommending several matching adjustments to the user, he needs to decide which one is best suited to his tasks. Hick's law [12] states that our decision time will increase logarithmically with the number of choices. When too many recommended matchings are displayed to the user, it would be difficult to choose one from them because of the abstract nature of matching. Visual cluttering is also likely if too many matchings are visualized. Our collaborator agrees that at most three top-ranked matchings should be recommended to the user. Other matchings with a lower rank may be visually summarized.

4.2.3 Matching Evaluation

After generating several matching recommendations, the trade-offs among the matchings should be visualized. We classify matching comparison into three levels and each provides different resolutions of insight into which matching is better:

Agent-level comparison In which matching does a particular agent get a better match (more preferable partner(s))? After improving the match of an agent, the user might consider the matching which gives this agent the best match the most preferable.

Comparison between two matchings When comparing matching A with matching B, which agents get a better match in A than in B and by how much is the match they get in A better? Which agents get a worse match in A and by how much? In determining whether matching A or matching B is better, the central planner needs to decide with his domain knowledge if the welfare of a particular agent should be improved or may be sacrificed.

Comparison amongst multiple matchings To compare multiple recommended matchings, the algorithmic result is first chosen as the baseline. For each recommended matching, how many agents have their welfare increased and how many have their welfare decreased, compared with the baseline (difference from the baseline)? Each recommended matching's differences from the baseline are then compared for the user to decide which recommended matching is a better improvement over the algorithmic result (difference-on-difference comparison).

Besides the three levels of matching comparison, the common performance metrics of different matchings (i.e. welfare, fairness and number of matched pairs) should also be visually summarized for evaluation.

4.3 System Workflow

The workflow in our matching adjustment model is shown in Fig 3. VisMatchmaker is a web-based application implemented in D3.js. Our system uses two data sets as the input: preference lists and a pre-allocation. To create the preference lists, the data obtained (e.g. the attributes of each agent) is first modelled in a way such that each agent has a list of the other agents. In the pre-allocation data set, each agent is represented by a tuple and the elements in a tuple are the partners of the agent. The workflow produces a new allocation as the output.

W1 The workflow starts with a matching which is the algorithmic result in the first iteration.

W2 The user first detects the problems in the algorithmic result. He might find a popular agent and want to assign a different agent to this popular agent or he might find some agents with a poor match and want to improve them.

W3 He then issues queries to steer the adjustment-searching algorithm which recommended matching adjustments to the user. If he issues one query, three top-ranked matchings satisfying the query are recommended. If he issues more than one query, the highest-ranked matching from each query is recommended to help the user decide which query gives rise to a better matching. For example, the user might find that assigning agent B to agent A is better than assigning agent C to agent A. He can issue up to three queries at a time to ensure the number of matching recommendations is limited to three.

W4 The matchings are then visualized by the system. The user can evaluate the result by comparing the matchings at three different resolutions and by comparing the summary statistics of the matchings.

W5 An iteration ends with the user chooses the matching he deems the best. The query associated with the chosen matching is stored. The matching chosen serves as an intermediate result and is used as the starting matching in the next iteration. When the user issues a new query in the next iteration, the stored query will be prepended to the new query to allow the user to adjust a matching progressively. For example, the user might have issued a query to assign agent A to agent B and chosen a matching satisfying the query in the first iteration. In the next iteration, it is likely that he wants A and B to be paired up already. The previous query of assigning A to B should therefore be prepended to the new queries.

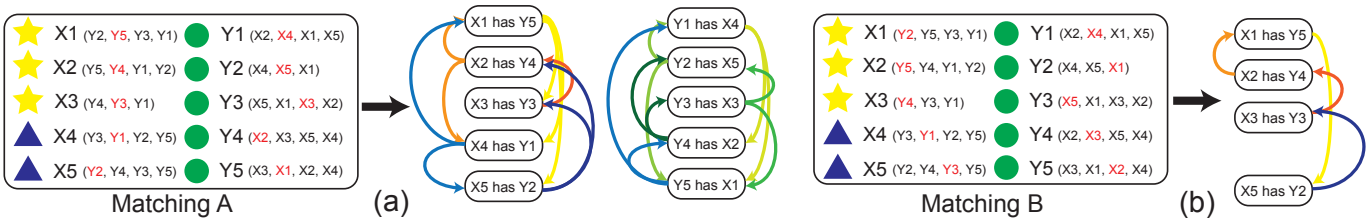

Fig. 4. A directed graph is first created for the original matching. Each cycle in the graph represents a way to swap partners with the others. The algorithm returns cycles based on the user's query.

5 Adjustment Searching

After the user issues queries through direct assignments and range searches, an algorithm searches for possible adjustments to the algorithm-generated matching and recommends them to the user. To satisfy the reasonable and comparable requirements, we use a variation of Top Trading Cycle Algorithm [25].

First, a directed graph is created for the algorithm-generated matching. The graph for matching A in Fig. 2. is shown in Fig. 4a. Each node in the graph represents an agent. In one-to-many and many-to-many matching problems, each node represents a quota of an agent. We say that agent B is acceptable to agent A if B is in A's preference list. A link from agent A to agent B means agent A considers agent B's current partner acceptable and agent B's partner finds agent A acceptable. For example, as X1 wants the partner of X3 (X3's partner (Y3) is in X1's preference list) and X3's partner (Y3) also wants X1, a link is directed from X1 to X3. Strongly connected components in the graph is then found by Kosaraju's algorithm [26] and the simple cycles are found by Johnson's algorithm [13]. Each simple cycle represents a way of swapping partners with each other. For example, matching B in Fig. 2 can be represented by the cycle shown in Fig. 4b. The cycle means X1 will get X5's partner (X1 points to X5), X5 will get X3's partner and so on. In one-to-many and many-to-many matching problems, some cycles may have different quotas of the same agent matched to the same partner and are therefore redundant. The redundant cycles are discarded in the recursive search. Cycles are found based on the queries and converted to matchings which are ranked and visualized.

The resulting matching is reasonable in the sense that the number of pairs is the same as that in the algorithmic result. Therefore, if the algorithmic result is reasonable (which is highly likely), the matching adjustments found are also reasonable. The depth of recursive search controls the number of agents in a cycle and hence the number of agents whose partners are changed. The comparable requirement is therefore satisfied by allowing the user to control the recursive depth.

6 VisMatchmaker

In this section, we first discuss the design rationale of our system. We then present the visual encodings and highlight the important interactive operations in each of these visualization techniques.

6.1 Design Rationale

Based on the task analysis in section 4, we further distilled a list of design requirements for developing a visual analytics system for exploring alternative matchings to the algorithmic result.

- R1 Revealing agents' popularity and their goodness of match for problem detection.
- R2 Providing interaction techniques to enable direct assignment and range search.
- R3 Facilitating the three levels of matching comparison.
- R4 Aggregating the important attributes of matchings (i.e. welfare, fairness and number of matched pairs) for matching evaluation.

6.2 Number Line View

The first design decision concerns the visualization of the goodness of each agent's match (R1). A number line visualizes different aspects of the goodness of an agent's match in multiple matchings (see Fig. 5) and

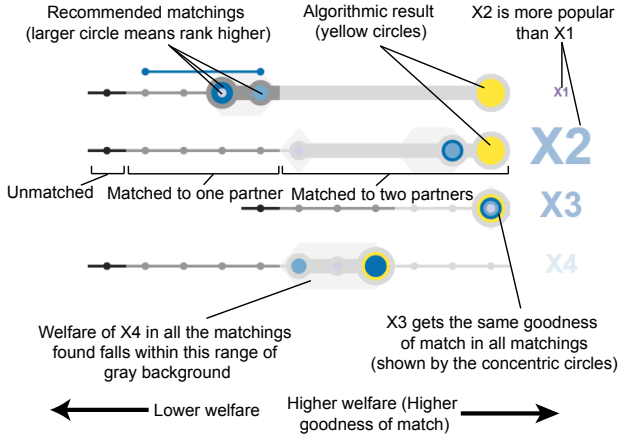

Fig. 5. A number line reveals different aspects of the goodness of match of the agent in different matchings.

hence it provides the agent-level comparison of matching (R3). The size of a label encodes the popularity of the agent (R1). An agent is more popular if more agents choose him as their top choices. The position of a colored circle on a number line indicates the goodness of the agent's match in a matching. Following the convention, moving from left to right along a number line, the welfare of the agent increases (the agent gets a better match). Hence, the leftmost circular mark indicates the worst outcome for an agent (unmatched) while the rightmost circular mark indicates his best possible match. When the maximum preference list length is four, the rightmost mark on the number line of an agent with 2 quotas indicates a match with his first two choices (welfare = 7), the second mark on the right represents a match to his first and third choices (welfare = 6), the third mark represents a match to his first and third choices or his second and fourth choices (welfare = 5) and so on.

A number line also reveals the number of filled quotas of an agent. Each number line is colored by several shades of gray which seperate the line into several regions (See Fig. 5). Each region represents the number of partner(s) the agent gets in a matching. For example, if an agent has two quotas, there are three regions in his number line. The leftmost, middle and the rightmost region means unmatched, matched to one partner and matched to two partners respectively.

As mentioned in section 4, three matchings are recommended to the user if there is only one query and the highest-ranked matching from each query is recommended to the user if there are more than one queries. These recommended matchings together with the initial algorithm-generated matching are explicitly visualized as circles on the number lines. Different colors are given to different matchings for identification. For example, the position of the yellow circles represents the goodness of the agents' matches in the algorithmic result. The relative rank of each matching are also visualized. The higher the rank of a matching (higher total welfare), the larger the circles of the matching. If the goodness of matches of an agent in is the same across multiple matchings, the circles of different colors are overlapped as concentric circles. The gray scale of a circle's border is the same as the color of the region on the number line in which the circle is located to help the user identify the number of an agent's partner(s) by simply looking at the circle. The portion of a number line between the rightmost circle and the leftmost circle are thickened. Its saliency tells the range of goodness of the agent's matches in multiple matchings and

helps the user understand more quickly if an agent has similar welfare scores across multiple matchings.

After issuing a query, more than three matchings may be found by the system. The lower-ranked matchings which are not displayed as circles are summarized by the gray backgrounds behind the number lines. The gray background on a number line notifies the user that in the pool of matchings which satisfy the current query, there exists a matching in which the goodness of the agent's match falls within the light gray background.

To give an overview of the number line view, when the user hovers on a circle, all the circles of same color are summarized in the summary view (Fig. 1b). Each horizontal line in the summary view represents a group of agent. The size of a circle at a particular position encodes the number of circles in the number lines which fall at that position. This helps the user understand the number of agents in each group who get their best possible match, their second best match and so on (R4).

6.3 Visual Preference List

There are three mutually exclusive states for an item in an agent's preference list and three colours are used to encode these states: 1) the agent is matched to the item (yellow), 2) the agent is not in the item's preference list and should not be paired up (white), 3) the agent is in the item's preference list but they are not matched (black). An agent's visual preference list (Fig. 1d) visualizes the states of the items in his preference list. Each cell represents an item in an agent's preference list and the cell color encodes its state. The leftmost cell is the best choice while the rightmost cell is the worst choice. The squares on the right of a visual preference list indicate the number of quotas of an agent which are filled (yellow) and not filled (black). The number of filled and unfilled quotas in each agent group are summarized in the summary view as bar charts (R4).

6.4 Stacked Graph View

The stacked graph view aims to support the comparison amongst multiple matchings (R3). The difference between a matching and the baseline matching is first computed and visualized as a stack. As shown in Fig. 6, a stacked graph structure is created for each of the recommended matchings other than the baseline. The stacks are then juxtaposed horizontally for easy comparison.

Consider a stack representing the comparison of a matching A and the baseline. Each small horizontal bar in the stack represents an agent. The rightmost edge of a bar represents the maximum welfare while the leftmost edge of the bar represents zero welfare (unmatched). The welfare scores of the agent in matching A and the baseline are marked. The two marks form a white rectangle. The length of the rectangle encodes the difference in welfare of the agent in the two matchings. The color of its border indicates the direction of change. Red indicates a higher welfare in A than in the baseline matching while green indicates a lower welfare. The small bars are stacked such that the marks representing the baseline matching form a vertical straight line. The right side of this vertical line therefore means a higher welfare in A. These bars are sorted by welfare difference so that the upper half represents a higher welfare in A and the lower half represents a lower welfare. The bars belonging to the same agent group are further grouped together and their background color represents their group.

The stacks of bars are then juxtaposed horizontally. The same group in different stacks are linked together with a band of the same color to help the user identify the numbers of agents in an agent group who are better-off or worse-off comparing with the baseline across multiple matchings. The stacks are aligned to a horizontal axis as shown in Fig. 6. In this way, the thickness of the upper half of a stack encodes the number of agents whose welfare is higher than in the baseline and the thickness of the lower half encodes the number of agents whose welfare is lower.

The circle label below each stack indicates which two matchings are compared in the stack. For example, the rightmost stack in Fig. 6 represents a comparison between the yellow and the dark blue matching. The default baseline matching is the original algorithm-generated result as the user is usually interested in knowing how the recommended matchings improve upon the original one. The baseline can be changed by clicking on a circle label. For example, by clicking the rightmost circle label, the baseline is changed to the dark blue matching.

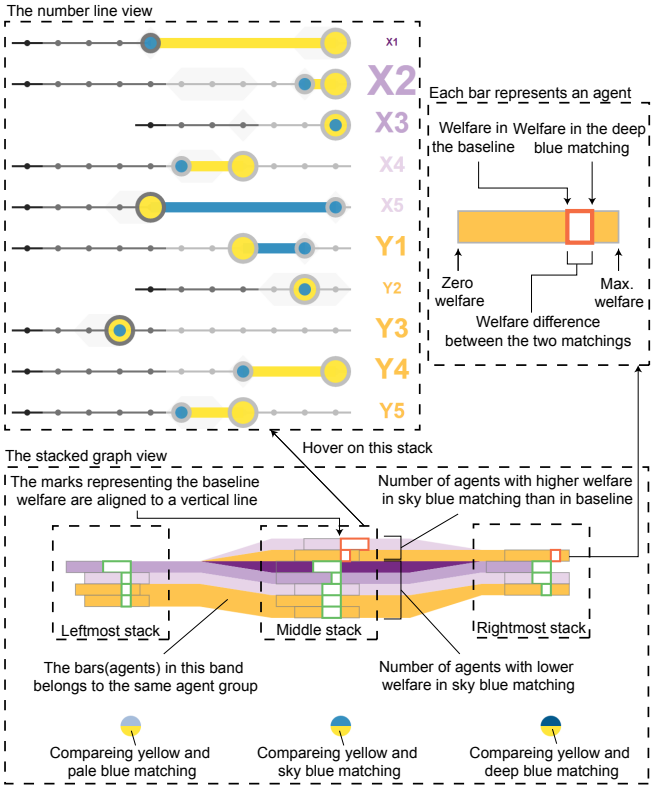

Fig. 6. The stacked graph view enables the comparison amongst multiple matchings. By hovering on a stack, the detailed differences between a matching and the baseline (the second level of matching comparison) is revealed in the number line view.

6.5 Welfare Glyph

Agents may be grouped hierarchically. We designed a circular glyph to display this hierarchical structure and summarize the welfare score of each group at each level (R4) (see Fig. 7). Each concentric circle represents a level in the hierarchy. A larger ring represents a lower level in the hierarchy so that the glyph is like viewing the tree from the top. The size of the innermost circle is mapped to the total welfare (value at the root of the tree). The welfare of a group is encoded by the length of an arc. The arcs are colored according to the groups they represent.

Fairness involves the comparison of welfare of different groups in the same level. This design allows the comparison of welfare of the groups within the same level by comparing the lengths of arcs in each level. It also enables the comparison of the welfare of a group in two matchings by comparing the lengths of the same arc in two glyphs.

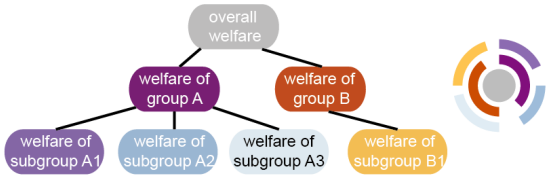

Fig. 7. The mapping of welfare scores to the welfare glyph.

6.6 Interactions

Different interaction techniques are implemented to support the workflow mentioned in Section 4. VisMatchmaker provides a tight integration of visualization and interaction which enables the user to directly interact with the visual interface. The interface of VisMatchmaker is shown in Fig. 1.

Problem detection (W2) The user is most interested in finding the popular agents and the agents with a poor match. The scatterplot provides an overview of the agents' popularity and goodness of match (R1). A window can be drawn in the scatterplot (Fig. 1e) to select the agents to be displayed as number lines in the number line view. The number line view supports scrolling for navigating the number lines.

Range search (W3) Upon identifying an agent with a poor match in the number line view, the user is enabled to improve it. The number line design provides an intuitive spatial metaphor for range search (R2). The user can simply drag to draw a line above a number line to indicate that an agent's match should fall within this range of goodness. By allowing range search on the number lines, interaction is tightly integrated with problem detection.

Direct assignment (W3) The user might want to change the assignment of a popular agent. The visual preference lists support direct assignment (R2). By hovering on the label of an agent, all the cells representing the agents in other agents' preference list are highlighted in red. For example, in Fig. 8, the label "X5" is hovered. It can be observed from the highlighted cells that Y2 is now matched to X5 (yellow), Y1 has chosen X5 as the fourth choice but X5 has not chosen Y1 (white) and Y3 and X5 are not matched (black). By clicking on a cell in a visual preference list, the agent is directly assigned to the selected choice. In Fig. 8, as indicated by the blue dots above Y3's first choice, the user requires Y3 to be paired with his first choice.

Setting multiple queries (W3) As mentioned in Section 4, multiple direct assignments and range searches can be combined to form a query. The user can set up at most three different queries at a time for comparing the highest-ranked matching from each query. The color of the circles in the number lines of the highest-ranked matching is the same as the color of the lines and dots of the query. The user can set one of the queries as an active query which is indicated by opaque lines above the number lines and opaque dots above the visual preference lists. Inactive queries are indicated by the transparency of the lines and dots. In Fig. 8, the blue query is "Y2 has two partners AND Y1 is paired with his first choice". The pink query is "Y2 has two partners AND Y3 is paired with his first choice". The opacity of the lines and dots reveals that the blue query is active and the red query is inactive.

The lower-ranked matchings from the active query are not visualized as colored circles but are summarized by the gray backgrounds behind the number lines. They guide the user to further refine a query by telling him what are in the complete pool of matchings found by the adjustment-searching algorithm. In Fig. 8, the gray background behind the number line of Y5 is on the right of the blue circle. This tells the user that in the pool of matching satisfying the blue active query, there are matchings which give Y5 a better match than that represented by the blue circle. The user can refine the blue query accordingly.

Matching evaluation (W4) The user then decides which matching recommended by the system is the best. The number line view provides the agent-level comparison while the stacked graph enables the comparison amongst multiple matchings. To delve into the detailed differences between a matching and the baseline (the second level of matching comparison), the user can hover on the stack representing the comparison of interest (R3). The agents in the stack are displayed as number lines in the number line view and the number line view will display only the circles representing the two matchings (see Fig. 6). In each number line, the thickened bar between the two circles is colored according to the matching in which the agent gets a better match. For example, a blue thickened bar in the number line of Y1 in Fig. 6 indicates that Y1 gets a better match in the blue matching and the length of the thickened bar represents the magnitude of the difference in Y1's goodness of match between the blue and the yellow matching.

The user may also want to compare the summary statistics of the recommended matchings and the algorithmic result. By hovering on a colored circle representing a matching in the number line view, the summary statistics of the matching are displayed in the summary view. The welfare glyph enables the comparison of welfare and fairness of different matchings while the bar chart in the summary view tells the number of matched agents and unfilled quotas in each agent group.

Next iteration (W5) The user may think one of the recommended matchings is the best among all but still want to refine it further. He can click on one of the circles in the number line view representing a matching. The matching chosen will then be set as an intermediate result and replace the algorithmic result for further refinement.

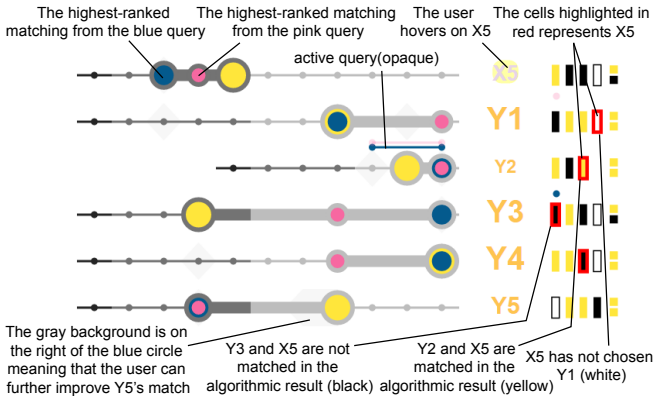

Fig. 8. The user hovers on X5 and the cells representing X5 are highlighted. Blue query: Y2 obtains a certain range of match and Y3 gets his first choice. Pink query: Y2 obtains a certain range of match and Y1 gets his first choice. The gray backgrounds provide a summary of the lower-ranked matchings satisfying the active blue query and guides the user to refine the blue query.

7 Evaluation

As usage of matching data requires consent of all the people involved and people in general do not want to reveal their preferences, despite the popularity of matching problems, it is virtually impossible to apply VisMatchmaker on real dataset solely for case study purpose. We are particularly interested in the effectiveness of the system in dealing with different degrees of conflict of interests (i.e. many agents chooses a particular agent). Conflict of interest is precisely why matching adjustment is needed. In a perfect world in which everyone's first choice is chosen only by him and his first choice also ranks him the first, algorithms will give everyone their first choice and human involvement is not necessary. After consulting our collaborator, we evaluate our design by modeling two cases: mild and serious conflict of interests.

7.1 Mild Conflict of Interests

The first use case demonstrates how a central planner match teaching assistants and professors using VisMatchmaker. Matching TAs and professors is considered a case of little conflict of interests. Very often, professors pick their graduate students as TAs and graduates students want to be the TAs of the course taught by their advisors. However, our department has a group of instructional assistants and teaching associates who also have TA duties. They are basically full-time TAs and are less involved in research. More TA duties should be allocated to them. We created a dataset with 30 professors and 70 teaching assistants (10 full-time TAs and 60 graduate students). Every professor needs two teaching assistants and every teaching assistant is responsible only for one course. Each person have 5 choices in their preference list. Each professor has two graduate students and both of them put their advisor as the first choice. The full-time TAs put random professors in their preference list. Each professor's students and the full-time TAs who rank them first will appear in the professor's list. The professors may rank the full-time TA higher than their students (full-time TAs are usually more experienced). The remaining empty slots of each person's preference list are random. The TAs and professors are matched using the most commonly used Deferred-Acceptance algorithm.

Let us assume that the goal of the central planner is to ensure all the full-time TAs are matched. After he loads the dataset, he observes that two of the full-time TAs (Shek and Mo in Fig. 9a) are not matched in the algorithmic result. With range search, he tries to ensure Mo is matched. The algorithm recommends three matchings to him and from the number line view, he finds that all three matchings give Mo his best possible match. As he observes that Shek is still not matched,

he attempts to improve it by issuing another range search for Shek. The matching recommended by the system changes accordingly as indicated by the changes of the colored circles in the number line view. He then finds that all the full-time TAs are matched in only one of the three recommended matchings (only the sky blue matching does not have any circles on the leftmost positions of the number lines which indicate unmatched).

Next, he wants to compare the sky blue matching and the algorithmic result (the yellow matching) by having an overview of them. By hovering on a sky blue circle and then a yellow circle in the number line view, the two matchings are summarized in the summary view. He finds that the total welfare of the pale blue matching is similar to that of the yellow matching as indicated by the similar center circles in the two welfare glyphs (Fig. 9b). The summary of the number line view shows that many professors get their first two choices in the sky blue matching. Moreover, all the quotas of the professors are filled meaning that all the professors get their required number of TAs (Fig. 9c). The central planner considers the sky blue matching best suited to his tasks.

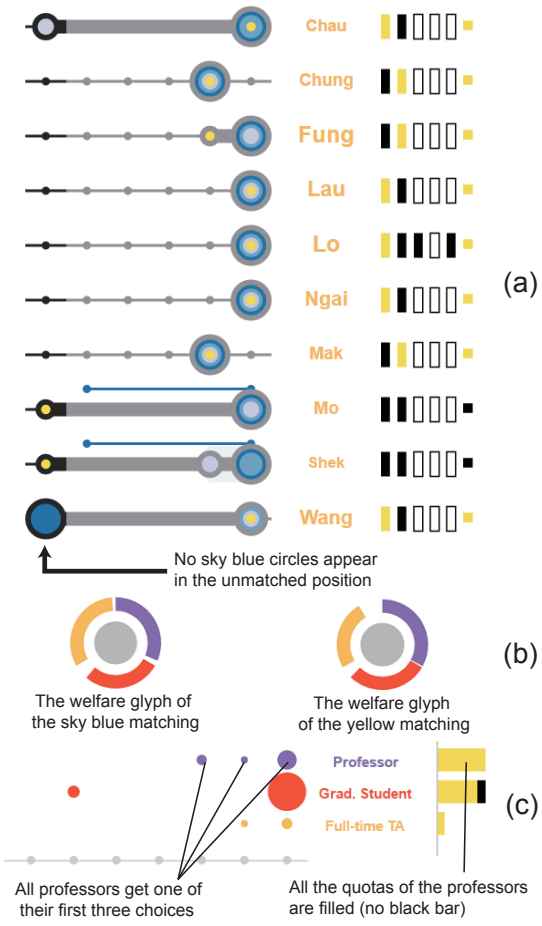

Fig. 9. (a) The number lines of the full-time TAs. (b) The welfare glyphs of the sky blue and the yellow matching. (c) The summary of the sky blue circles in the summary view.

7.2 Serious Conflict of Interests

The second use case concerns how a central planner matches professors and courses. Choosing courses to teach can be highly competitive at times. Professors with the same research area may be interested in teaching the same course related to their expertise. Also, a number of elementary courses which do not require specialized expertise are likely to be chosen by many professors. Hence, we model it as a case of serious conflicts of interests. In our synthetic dataset, there are 20 courses (10 elementary courses and 10 advanced courses which require more specialized knowledge) and 30 professors. There are 5 choices in the preference lists of both professors and courses. Every three professors have the same research area and choose the same advanced course as their first choice. They randomly pick any elementary courses as the second, third, fourth and fifth choices. For the preference list of a course, after collecting the lists of professors who have chosen it, the list is truncated to 5 choices to remove some professors who do not pick it as the first choice. This is to simulate the fact that 1) professors who have the expertise teaching an advanced course and pick the course as their first choice should be more suitable to teach the course and should not be removed, and 2) professors who get a low rating when he taught the course or who have not taught the course before are removed from consideration. Again, Deferred-Acceptance algorithm is used to match the professors and the courses.

Let us consider a situation in which the central planner received some special requests from a number of professors concerning their teaching preferences. Professor A and B both want to teach "Computer Graphics". Professor C wants to teach in this semester. As a first task, the central planner points to the name "Computer Graphics" in the number line view and observes that the course is assigned to Professor D. He tries to issue two queries to see what will happen to the matching if the course is assign to A or B. In the first query, the course is assigned to A by direct assignment. In the second query, the course is directly assigned to B.

He then compares the best matching from each query. By looking at the stacked graph, he finds that compared with the algorithmic result, only two professors are affected if the course is assigned to B (left stack in Fig. 10a). However, if the course is assigned to A, quite a number of professors are affected and many of them have their welfare decreased (right stack in Fig. 10a). He decides to assign the course to B and clicks on a pink circle (which belongs to the best matching from the second query) in the number line view to set it as an intermediate result.

In the next iteration, he handles Professor D's request. He finds from the number line view that D is not matched in the intermediate result. On top of the previous query to assign "Computer Graphics" to Professor B, he tries to ensure D is matched by range search. He observes from the number line view that in a recommended matching (the sky blue matching), Professor E who also wants to teach in this semester is unmatched. Therefore, he considers only the other two matchings recommended (the deep blue and the pale blue matching). He found from the stacked graph that the deep blue and the pale blue matchings are very similar in terms of the number of professors whose welfare is increased and the number of those whose welfare is decreased comparing with the intermediate result (the yellow matching) (leftmost and rightmost stack in Fig. 10b). By changing the baseline matching to the pale blue matching, he finds that the deep blue matching differs from the pale blue matching by only two professors (the rightmost stack in Fig. 10c). One gets a better match while another gets a worse match. He is indifferent between the two matchings and decides to choose the one which is ranked first by the algorithm (the dark blue matching).

7.3 Expert Interview

To evaluate the effectiveness of our design, we had interviews with our collaborator and Prof. G. Prof. G was the UG coordinator in a university department and had several years of experience assigning professors to university courses. We demonstrated the system workflow to them in two one-on-one interviews and collected their feedback.

In the interview with our collaborator, he raised several possible scenarios while matching professors and TAs and we demonstrated to him how our system can solve his problems. He was impressed by several features provided by the system. First, he commented that the system allowed him to adjust a matching and the number lines intuitively showed him the goodness of each person's match through position. He also mentioned that in the past, they could only use spreadsheet to do matching and adjust a matching. He said that it was very hard to know who gets a better match and who gets a worse match after an adjustment with spreadsheet. Second, he thought that the function to compare different queries is very useful. The third features he found good about our system was that our system offers multiple matching recommendations to the user. He thought that the

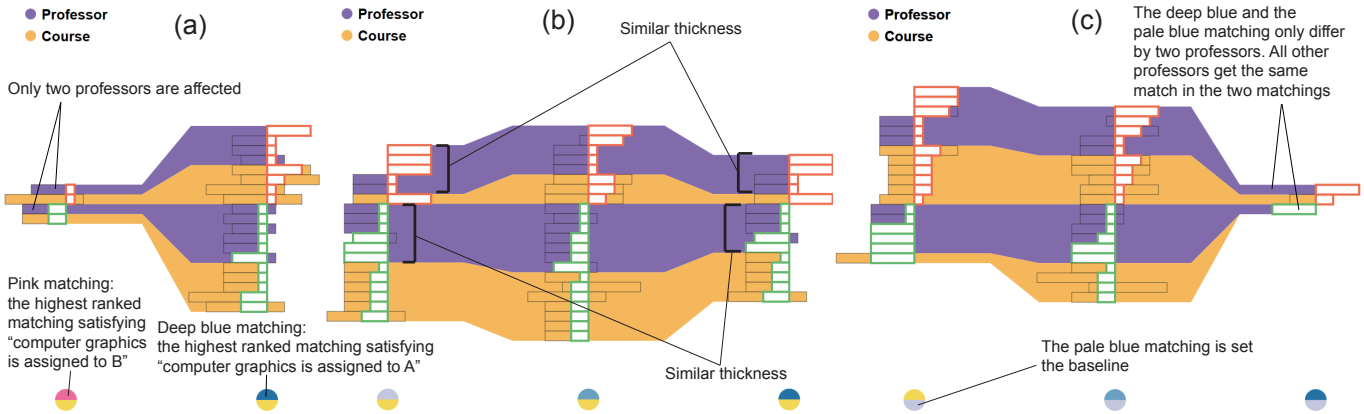

Fig. 10. Using stacked graphs for matching comparison

differences of the three recommendations are clearly visualized through the stacked graph view. Finally, concerning the interactive operations, he highlighted that it was a good idea to allow the user to drag on a number line to perform range search. On the whole, he said that "the system gives him more confidence with the assignment of TAs to professors and the whole process becomes smoother".

However, our collaborator was concerned that it may take a long time to learn how to use the system. One issue he raised was that the system does not show directly to whom an agent is matched. Besides that, he also expressed concerns over design of visual preference lists. He noted that when there are a lot of items in an agent's preference list, his visual preference list will be long and take up a large screen space.

For the interview with Prof. G, we were particularly interested in knowing how our system could help with his tasks while assigning professors to courses. We first asked him about the assignment process. He told us that every professor was required to rank six courses they want to teach and submit the list to him in a spreadsheet. He normally did it manually by first handling the courses which were signed up by the least number of professors. The courses which were signed up by many professors were handled the last. When we described the system workflow to him, he commented that the system would be quite useful and confirmed that the popularity of courses would be something he wanted to know while handling a matching. Besides, he thought that matching adjustment was a complicated process in that a local adjustment would easily affect the whole matching and allowing the user to understand the "change of the global picture upon a local adjustment" was the strength of our system. When we asked him what interested him the most when comparing two matchings, he replied that he wanted to know "the degree of mismatch of the two matchings" which was basically some kinds of summary statistics. However, he agreed that the three level of matching comparison would be a useful extension to summary statistics alone as "they allow the user to understand the reasons for the differences in the summary statistics of the two matchings".

8 Discussion

Our workflow and visual designs including the number line view and the stacked graph view can be extended to other matching problems. There are many problems in which each agent has a list of the other agents which are ranked by criteria other than preference. They may be ranked by criteria such as suitability (e.g. a reviewer has a list of papers ranked according to his expertise). With such a list of each agent and a pre-allocation, our system can be used for refining the assignment.

To address scalability issues when there is a huge number of agents, our system is designed based on the mantra "overview first, zoom and filter, then details-on-demand." [28]. There are three sources of potential scalability issues: problem detection, matching recommendation and matching evaluation. For problem detection, the scatterplot provides an overview of the popularity and the goodness of match of the agents. The user can filter and select the agents to be displayed as number lines in order to manipulate the matching. One deficiency of our proposed algorithm for matching recommendation is that it can be slow and affect user experience when the number of agents is large. To shorten the computational time, our system enables the user to set an upper bound on the number of matchings found by the system and the number of agents whose welfare is changed after the adjustment. Another approach is to find all the cycles from the directed graph before the adjustment begins. However, it may take a long time for pre-processing and deters the user from using the tool for matching adjustment. For matching evaluation, the stacked graph view provides an overview of the differences between the matching recommendations. To delve into the details, the user can hover on a stack of interest.

Finally, as mentioned by our collaborator, while the visual preference lists can show the states of each item in a preference list clearly, it will be long when the preference list of an agent has many items. Furthermore, it does not show the name(s) of an agent's partner(s) directly. While the name of the agent represented by a cell in a visual preference list can be revealed through interactions (e.g. by hovering on a cell, the name of the agent is displayed), interactions increase user's navigation time. These problems can be solved by allowing the user to switch between the visual preference lists and another mode of display. This mode should 1) show directly the name of the partner(s) to whom an agent is matched on the right of each label in the number line view, 2) visualize the rank of an agent's partner(s) in his preference list and 3) replace direct assignment by allowing the user to click on a label to display a list of the other agents that the user may want to directly assign to the selected agents.

9 Conclusion and Future Work

We have presented VisMatchmaker, a visualization system which helps the central planner explore alternatives to an algorithmic result based on his domain knowledge. The system enables the user to detect problems in a matching. We provide two kinds of algorithmic supports for the user to steer the adjustment-searching algorithm to compute the influence of an adjustment to the whole matching. Matching recommendations are provided to the user and the user can compare matchings at different resolutions with our visual designs. Our case studies show that our system is effective in dealing with real life scenarios.

In the future, we plan to further improve our system based on the feedback from our collaborator, inspect more varieties of matching problems and extend the current techniques to these problems. We would like to motivate the research on better adjustment-searching algorithms which can search for the matchings required by the user more efficiently. We also intend to collect feedback from potential users of the system and conduct experiments to improve its design.

Acknowledgments

We would like to thank anonymous reviewers for their valuable comments and Gary Chan for expert interview. This project is partially funded by RGC GRF16208514.

REFERENCES

[1] Mathsite: Stable marriage problem. http://mathsite.math.berkeley.edu/smp/smp.html. Accessed: 2016-06-19.

[2] D. Abraham. Algorithmics of two-sided matching problems. *Master's thesis, University of Glasgow, Department of Computing Science*, 2003.

[3] M. Ankerst, M. Ester, and H.-P. Kriegel. Towards an effective cooperation of the user and the computer for classification. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 179–188. ACM, 2000.

[4] E. Anshelevich and S. Das. Matching, cardinal utility, and social welfare. *ACM SIGECom Exchanges*, 9(1):4, 2010.

[5] C. Barnhart, A. M. Cohn, E. L. Johnson, D. Klabjan, G. L. Nemhauser, and P. H. Vance. Airline crew scheduling. In *Handbook of transportation science*, pp. 517–560. Springer, 2003.

[6] Z. Feng and X. Ruihua. Study on model and algorithm for urban rail transit crew scheduling system. In *International Conference on Computer Design and Applications (ICCDA)*, vol. 5, pp. V5–213–V5–216, June 2010.

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

[8] D. Gale and M. Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11(3):223–232, 1985.

[9] M. Ghoniem, J. D. Fekete, and P. Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *IEEE Symposium on Information Visualization (INFOVIS)*, pp. 17–24, 2004.

[10] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, 2011.

[11] C. Haas. *Incentives and Two-Sided Matching-Engineering Coordination Mechanisms for Social Clouds*, vol. 12. KIT Scientific Publishing, 2014.

[12] W. E. Hick. On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, 4(1):11–26, 1952.

[13] D. B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 4(1):77–84, 1975.

[14] S. O. Kimbrough and A. Kuo. On heuristics for two-sided matching: Revisiting the stable marriage problem as a multiobjective problem. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pp. 1283–1290. ACM, 2010.

[15] D. Liu, A. P. Sprague, and J. G. Gray. Polycluster: an interactive visualization approach to construct classification rules. In *International Conference on Machine Learning and Applications (ICMLA)*, pp. 280–287, 2004.

[16] J. Liu and D. M. Chiu. Reciprocating preferences stablize matching: College admissions revisited. *arXiv preprint arXiv:1011.1135*, 2010.

[17] T. Liu, J. Ma, W. Guan, Y. Song, and P. Fu. Design and implementation of bus crew scheduling system using integrated case-based and rule-based reasoning. In *Fifth International Joint Conference on Computational Sciences and Optimization (CSO)*, pp. 475–479, June 2012.

[18] J. S. Malasky. *Human machine collaborative decision making in a complex optimization system*. PhD thesis, Massachusetts Institute of Technology, 2005.

[19] T. Munzner. Why: task abstraction. In T. Munzner, ed., *Visualization analysis & design*, chap. 3, pp. 43–65. CRC Press, Boca Raton, FL, 2014.

[20] M. Nakamura, K. Onaga, S. Kyan, and M. Silva. Genetic algorithm for sex-fair stable marriage problem. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 1, pp. 509–512, 1995.

[21] A. Perer and B. Shneiderman. Balancing systematic and flexible exploration of social networks. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):693–700, 2006.

[22] L. Pizzato, T. Rej, J. Akehurst, I. Koprinska, K. Yacef, and J. Kay. Recommending people to people: the nature of reciprocal recommenders with a case study in online dating. *User Modeling and User-Adapted Interaction*, 23(5):447–488, 2013.

[23] R. Sadana, T. Major, A. Dove, and J. Stasko. Onset: A visualization technique for large-scale binary set data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1993–2002, 2014.

[24] J. Sethuraman, C.-P. Teo, and L. Qian. Many-to-one stable matching: geometry and fairness. *Mathematics of Operations Research*, 31(3):581–596, 2006.

[25] L. Shapley and H. Scarf. On cores and indivisibility. *Journal of mathematical economics*, 1(1):23–37, 1974.

[26] M. Sharir. A strong-connectivity algorithm and its applications in data flow analysis. *Computers & Mathematics with Applications*, 7(1):67–72, 1981.

[27] A. S. Shibghatullah, T. Eldabi, and G. Rzevski. A framework for crew scheduling management system using multiagents system. In *28th International Conference on Information Technology Interfaces*, pp. 379–384, 2006.

[28] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *IEEE Symposium on Visual Languages*, pp. 336–343, 1996.

[29] S. Van Den Elzen and J. J. Van Wijk. Baobabview: Interactive construction and analysis of decision trees. In *IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 151–160, 2011.

[30] S. van den Elzen and J. J. van Wijk. Multivariate network exploration and presentation: From detail to overview via selections and aggregations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2310–2319, 2014.

[31] T. Yamada, T. Sato, T. Tomiyama, and N. Ueki. Real-time railway crew rescheduling: Performance support with explanations. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1005–1010, Oct 2015.

[32] J. Zhao, Z. Liu, M. Dontcheva, A. Hertzmann, and A. Wilson. Matrixwave: Visual comparison of event sequence data. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 259–268. ACM, 2015.