Least-Unpopularity-Factor Matching

Richard Matthew McCutchen
Montgomery Blair High School
http://www.kepreon.com/~matt/

Advisor:
Dr. Samir Khuller
University of Maryland
http://www.cs.umd.edu/~samir/

Summary:

One of the most common administrative tasks that organizations perform is the assignment of people to positions based on the people's preferences. A computer program that chooses an assignment automatically would be useful, but what assignment rule should it follow? I proposed the rule of least-unpopularity-factor matching, which is difficult to subvert and always designates a "best" assignment—but I proved that there is no practical way to actually compute that assignment for large numbers of people and positions. The result eliminates one possibility in the search for a rule suitable for an automatic assignment program but suggests where a suitable rule might be found.

1 Introduction

One of the most common administrative tasks that an organization performs is the assignment of people to positions based on preferences expressed by the people, the positions, or both. For example, the University of Maryland Department of Computer Science assigns teaching assistants to classes according to the teaching assistants' preferences. The National Resident Matching Program (see [9]) assigns residents to hospitals based on the preferences of both residents and hospitals. Gale and Shapley [3] even suggested that students could be assigned to colleges by a central organization that observes the preferences of both sides.

All of these organizations face the problem of finding an assignment that gives fair consideration to everyone's preferences. We consider one version of the problem in this paper. Suppose we have a set of people and a set of positions, each of which can hold one person. Each person provides a *preference list* ranking the positions he or she is willing to accept; there may be ties in the ranking. A central authority collects the lists and, for each person P, either assigns P to a position on P's list or leaves P without a position. The preferences are *one-sided* because the people rank the positions; if the positions also ranked the people, the preferences would be *two-sided*.

In which position should the authority place each person? No assignment is clearly identifiable as the best because any assignment will inevitably please some people at the expense of others. It isn't even obvious which of two assignments should be considered better if some people are better off in the first while others are better off in the second.

When a human constructs an assignment, he or she generally eyeballs all the preferences and arbitrarily starts filling positions. This approach has several disadvantages: it is time-consuming and error-prone when there are many people and positions, and the human may favor some people over others, even subconsciously. A computer program that performs the assignment automatically would be a great convenience. Furthermore, any possibility of favoritism could be ruled out if the program used an objective rule to select the "best" assignment. However, as we mentioned, there is no obvious rule to use.

One rule, known as *popular matching*, was proposed by Gärdenfors [4] and studied further by Abraham et al. [1]. According to popular matching, the authority should select an assignment M such that there is no other assignment N that the people would prefer to M by a majority vote, where each person votes for the assignment that gives him or her the better position or abstains if he or she receives equally good positions in both. Such an M is called *popular*.

For example, consider a case with three people denoted A, B, and C, three positions denoted w, x, and y, and the following preference lists:

$$\begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} w & x & y \end{array} \\ \begin{pmatrix} 1 & - & 2 \\ 1 & 2 & - \\ - & 1 & 2 \end{pmatrix} \end{array}$$

This notation means that A's first choice is w and her second choice is y; she will not accept x. One possible assignment M_1 is given by the underlined preference-list entries:

$$M_1 = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} w & x & y \end{array} \\ \begin{pmatrix} \underline{\mathbf{1}} & - & 2 \\ 1 & \underline{\mathbf{2}} & - \\ - & 1 & \underline{\mathbf{2}} \end{pmatrix} \end{array}$$

M_1 is not popular because more people would vote in favor of switching to the following alternative assignment N_1 than would oppose the switch (specifically, B and C would outvote A):

$$N_1 = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} w & x & y \end{array} \\ \begin{pmatrix} 1 & - & \underline{\mathbf{2}} \\ \underline{\mathbf{1}} & 2 & - \\ - & \underline{\mathbf{1}} & 2 \end{pmatrix} \end{array}$$

In fact, N_1 is popular; it is straightforward to verify that there is no other assignment to which the people would switch by majority vote.

In this small example we could identify a popular assignment directly from the definition, but this approach is impractical for larger cases because the number of possible assignments is exponential in the number of people

and positions. It would take much too long to consider each assignment M in turn and determine whether it is popular; in fact, it would take much too long to consider even a single M if we compared it to each alternative assignment N. Fortunately, the authority can use the polynomial-time algorithm discovered by Abraham et al. [1] to quickly find a popular assignment.

Intuitively we would expect every set of preference lists to admit at least one popular assignment because, if the people keep voting in favor of improvements to an assignment, they should eventually reach an assignment on which no further improvement is possible. Surprisingly, this is not always the case: nontransitivity may arise in the voting and a popular assignment may fail to exist. For example, consider these preferences:

$$\begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} w & x & y \end{array} \\ \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix} \end{array}$$

This assignment M_2 seems to be the best we can do:

$$M_2 = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} w & x & y \end{array} \\ \begin{pmatrix} \underline{\mathbf{1}} & 2 & 3 \\ 1 & \underline{\mathbf{2}} & 3 \\ 1 & 2 & \underline{\mathbf{3}} \end{pmatrix} \end{array}$$

However, M_2 is not popular because the people would vote in favor of switching to N_2 (B and C outvoting A):

$$N_2 = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} w & x & y \end{array} \\ \begin{pmatrix} 1 & 2 & \underline{\mathbf{3}} \\ \underline{\mathbf{1}} & 2 & 3 \\ 1 & \underline{\mathbf{2}} & 3 \end{pmatrix} \end{array}$$

Neither is N_2 popular, since C and A would outvote B in favor of switching to yet another assignment.

This is an example of a case in which the popular matching rule is useless because no assignment is popular. Abraham et al. [1] present empirical evidence that these cases are rare when the preference lists are independently random. However, the assumption of independently random preference lists is unrealistic in the real world: people will often have similar beliefs about which positions are more desirable than others. In fact, it was exactly this effect that led to the nontransitive voting in our example. We can expect popular matching to be similarly inapplicable to a significant fraction of the preference-list sets that would arise in the real world.

In view of this defect in popular matching, we might consider abandoning it completely in favor of a different assignment rule. However, most other rules can be subverted, meaning that a person can obtain a better position by lying about his or her preferences. For example, MIT assigns incoming freshmen to dormitories automatically based on preference lists they submit, and a freshman can improve his or her chances of being assigned to his or her true favorite dormitory by inserting highly desirable dormitories near the top of his or her preference list [8]. Popular matching is immune to manipulations of this nature because it considers only the relative order of positions on each person's preference list (through voting), not the positions' numerical ranks. Intuitively, if a person adds a position to his or her list, he or she might get assigned to it, but the new position will not affect the relative merits of assigning him or her to previously listed positions.

To be appropriate for general use, an assignment rule should be difficult for people to subvert, and we are most likely to obtain such a rule by repairing the defect in popular matching. We must help the authority distinguish between "bad" and "terrible" assignments in the event that no "good" (i.e., popular) assignment exists.

To this end, I will introduce the *unpopularity factor* of an assignment, which is a numerical measure of its "badness". I propose that the authority should select the assignment with the least unpopularity factor (or one such assignment if there is a tie); I call this rule *least-unpopularity-factor matching*. Least-unpopularity-factor matching is a universally applicable assignment rule that is difficult to subvert, and it is one of the first such rules ever proposed. In fact, when at least one popular assignment exists, the assignments of least unpopularity factor are exactly the popular assignments; in this sense, least-unpopularity-factor matching generalizes popular matching.

I will then develop the theory of least-unpopularity-factor matching, leading to two main results. First, I will present an efficient algorithm to find the unpopularity factor of a given assignment (Algorithm 4.4). When there

are very few people and positions (no more than about 10 each), one can find an assignment of least unpopularity factor exhaustively by running Algorithm 4.4 on every possible assignment. Second, I will prove that the problem of finding an assignment of least unpopularity factor is *NP-hard* (Corollary 5.4), meaning that no efficient way to find such an assignment for large numbers of people and positions is likely to be discovered. The search for a universally applicable assignment rule that is difficult to subvert and admits an algorithm to find the designated assignment, even for large numbers of people and positions, must continue.

1.1 Preliminaries

In the remainder of the paper, we will follow standard terminology and refer to assignments as *matchings*, positions as *posts*, and a setup consisting of people, posts, and preference lists as an *instance* of the matching problem with one-sided preferences. Like Abraham et al. [1], we will eliminate a special case by introducing for each person p a separate *last resort* post $\mathrm{LR}(p)$ available only to p as p's least favorite post. Assignment to the last resort represents the lack of an acceptable post assignment.

2 Related Work

The notion of popular matching was first introduced by Gärdenfors [4] for two-sided preferences. More recently, Abraham et al. [1] discussed popular matchings based on one-sided preferences and gave the first polynomial-time algorithm to find such matchings. Their algorithm runs in $O(n+m)$ time for preference lists without ties, where n is the number of people and m is the total number of preference-list entries. They also gave an algorithm that runs in $O(\sqrt{n}m)$ time for preference lists that may have ties. Abraham and Kavitha [2] suggested that the authority's behavior when no popular matching exists could be based on the graph induced by the "beats by majority vote" relation among matchings. In this paper, we take a different approach by generalizing the "beats" relation itself.

Rank-maximal matching is another rule for instances with one-sided preferences. A matching M is *rank-maximal* if it has the lexicographically maximum tuple $(n_1, n_2, \ldots)$, where n_i is the number of people assigned to ith choice positions. In other words, M must assign the maximum number of people to first choices; of all matchings with that property, M must assign the maximum number of people to second choices, and so forth. Irving et al. [6] found an algorithm that computes a rank-maximal matching in $O(\min(n+C, C\sqrt{n})m)$ time, where C is the worst rank to which any person is assigned (e.g., 3 for a third choice).

If the people assign numerical values instead of ranks to the positions, the most natural rule is to maximize the total value of all pairs; it is known as *weighted matching* [7]. In instances with two-sided preferences, we think of people as applicants and positions as recruiters. The most commonly used rule for these instances is *stable marriage*. It arises from a simulation in which a mutually consenting applicant and recruiter can pair themselves, abandoning their previous partners. Thus, a matching is considered *stable* if no currently unpaired applicant and recruiter would both be better off after pairing themselves. Gale and Shapley [3] proposed the rule, proved that a stable matching always exists, and gave an algorithm to find one.

A matching is *Pareto efficient* if no change is possible that improves the situation of at least one person without making anyone worse off; such a change would be a *Pareto improvement*. Pareto efficiency is a necessary condition for all reasonable kinds of optimality.

3 Least-unpopularity-factor matching

Definition 3.1. The *unpopularity factor* of a matching M is the maximum of u/v over all matchings N for the same instance, where u is the number of people who strictly prefer N to M and v is the number of people who strictly prefer M to N. Matchings N for which $u = v = 0$ are ignored. In other words, the unpopularity factor of a matching gives the maximum number of people who could improve for each person made worse off if another matching were adopted. □

It is possible that, for some N, $v = 0$ but $u > 0$, meaning that M has unpopularity factor ∞; in this case, N is a Pareto improvement over M. Clearly a matching is Pareto efficient if and only if its unpopularity factor is

finite. Furthermore, a matching is popular as defined by Abraham et al. [1] if and only if its unpopularity factor is 1 or less.

The *least unpopularity factor* of an instance is the minimum of the unpopularity factors of all its matchings, and the matching(s) that achieve this minimum are *as popular as possible*. Thus, the number of matchings that are as popular as possible can vary, but there is always at least one.

Let us revisit the example instances and matchings discussed in the Introduction. M_1 has unpopularity factor 2 for the first instance because N_1 achieves $u/v = 2$; it is clear by inspection that no other alternative matching achieves a greater u/v. However, N_1 has unpopularity factor 1: the only person who could possibly want to switch matchings is A, and to promote A we must demote B to his last resort, achieving $u/v = 1$. In the second instance, the six matchings that fill all three posts (including M_2 and N_2) all have unpopularity factor 2 because the people in x and y can improve at the expense of the person in w. No matching has a smaller unpopularity factor, so these six matchings are as popular as possible for the second instance.

3.1 Unpopularity factors and alternating paths

So far, we have shown matchings and instances as preference tables. However, we can also think of an instance as a bipartite graph whose vertices represent people and posts and whose edges represent preference-list entries. In this sense, a matching is a subset of the edges such that each person is incident on exactly one edge in the subset and each post is incident on at most one edge. Figure 1 shows the preference table and bipartite graph representations of the matchings M_1 and N_1 from the Introduction.

$$M_1 = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} w & x & y \end{array} \\ \begin{pmatrix} \underline{\mathbf{1}} & - & 2 \\ 1 & \underline{\mathbf{2}} & - \\ - & 1 & \underline{\mathbf{2}} \end{pmatrix} \end{array}$$

$$N_1 = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} w & x & y \end{array} \\ \begin{pmatrix} 1 & - & \underline{\mathbf{2}} \\ \underline{\mathbf{1}} & 2 & - \\ - & \underline{\mathbf{1}} & 2 \end{pmatrix} \end{array}$$

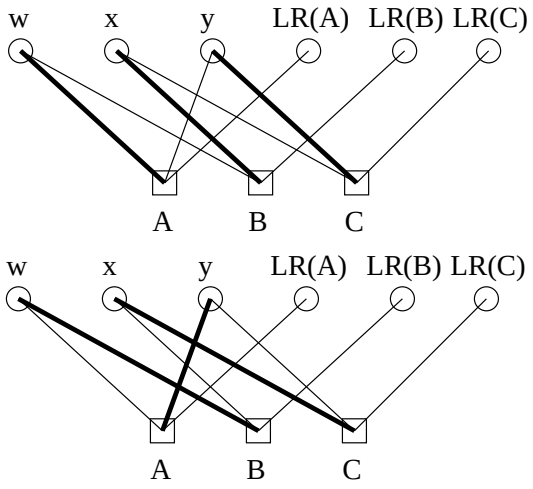

Figure 1: M_1 and N_1 shown as preference tables and bipartite graphs.

For matchings M and N, let $M \oplus N$ denote the symmetric difference of M and N. $M \oplus N$ consists of disjoint paths and cycles that are alternating in the sense that the edges come alternately from M and N. All people are matched in both M and N (even if to last resorts). If an alternating path of $M \oplus N$ stopped at a person, that person would be matched in M or N but not both, which is impossible; thus alternating paths of $M \oplus N$ end at posts. We can think of an alternating path as a sequence of reassignments: one person leaves her current post to fill an empty post, then someone else enters the post she left, etc. Each reassignment makes a person better off or worse off or neither if the two posts were tied on the person's preference list. An alternating cycle is similar except that all reassignments must "happen at once".

Conversely, if we have a collection X of disjoint paths and cycles that are alternating with respect to a matching M, we say that X is *applicable* to M and that *applying* the paths and cycles to M yields the matching $M \oplus X$.

Figure 2 shows the graph $M_1 \oplus N_1$ for the matchings M_1 and N_1 discussed in the Introduction. It consists of one alternating cycle with three reassignments: A is demoted from w to y, B is promoted from x to w, and C is promoted from y to x.

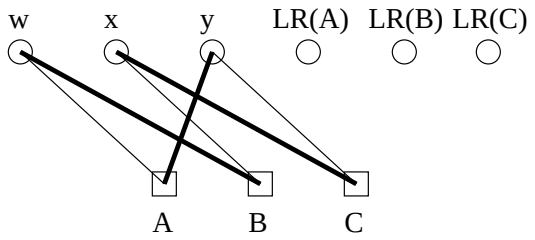

Figure 2: The graph $M_1 \oplus N_1$. Edges from N_1 are bold.

Though the unpopularity factor of a matching M is defined by considering all alternative matchings N, one can determine the unpopularity factor by considering only matchings N that result from applying a single alternating path or cycle to M. This idea is developed in Theorem 3.3 below.

Definition 3.2. A *basic Pareto improvement* over a matching M is an alternating path or cycle applicable to M that improves at least one person without making anyone worse off. If k is a nonnegative integer, a *basic pressure-k improvement* over M is an alternating path applicable to M that improves exactly k people and demotes exactly one person p by sending p to his last resort. (The reason for the term "pressure-k" will become clear in section 4.) □

Theorem 3.3. Let M be a matching. The unpopularity factor of M is ∞ (i.e., M is Pareto inefficient) if and only if there exists a basic Pareto improvement over M. Otherwise, the unpopularity factor of M is the largest integer k for which there exists a basic pressure-k improvement over M.

Proof: If a basic Pareto improvement (in fact, any Pareto improvement) over M exists, the unpopularity factor of M is clearly ∞. Conversely, if the unpopularity factor of M is ∞, there exists a matching N that is a Pareto improvement over M. At least one path or cycle in $M \oplus N$ must improve someone, and that path or cycle is a basic Pareto improvement over M.

Now suppose the unpopularity factor of M is a finite integer f. No basic pressure-k improvement over M can exist for any $k > f$. For if one did, applying it to M would yield an alternative matching that gives $u/v = k/1 = k > f$ in Definition 3.1, but f is the maximum u/v over all alternative matchings to M, a contradiction.

To complete the proof, we must construct a basic pressure-f improvement over M. Let N be the alternative matching to M that gives the maximum value of u/v in Definition 3.1, namely f. Let N_1 be the matching that is identical to N except that each person who is assigned a worse post in N than in M is assigned instead to her last resort in N_1. Clearly there are still u people better off and v people worse off in N_1 than in M.

Let $X = M \oplus N_1$. Let $X_1, \ldots, X_c$ be the paths and cycles constituting X. Let u_i be the number of people who are better off in $M \oplus X_i$ than in M and let v_i be the number of people who are worse off. We have $\sum_i u_i = u$ and $\sum_i v_i = v$. By the Pigeonhole Principle, there exists an i for which $u_i \geq u/c$.

If $v_i = 0$ for some i, then X_i is a basic Pareto improvement over M, so the unpopularity factor of M is ∞, a contradiction. Thus $v_i \geq 1$ for all i, so $\sum_i v_i \geq c$, so $v \geq c$, so $u/c \geq u/v$. Combining this with the result of the previous paragraph, there exists an i for which $u_i \geq u/v = f$.

Since $v_i \geq 1$, X_i, if applied to M, makes some person p worse off. By construction of N_1, p is demoted to his last resort from a better post. X_i has an edge of N_1 connecting p to his last resort, and no one other than p can ever be matched to that last resort. Thus, X_i must be a path that ends with the edge of N_1 connecting p to his last resort. If X_i also made another person q worse off, we could apply the same argument, but only one end of X_i is an edge of N_1; thus p is the only person made worse off by X_i.

We have shown that X_i is a basic pressure-u_i improvement over M. We showed $u_i \geq f$ and argued earlier that no basic pressure-k improvement over M could exist for $k > f$, so X_i is a basic pressure-f improvement over M and the result follows. □

Corollary 3.4. The unpopularity factor of any matching, if finite, is an integer. □

Now we have a first attempt at an approach to finding the unpopularity factor of a matching M: search for basic pressure and Pareto improvements over M. An efficient way to find the unpopularity factor will be developed in the next section.

4 Analyzing matchings using pressures

Definition 4.1. Let M be a matching that admits no Pareto-improving alternating cycle. The *pressure* of a post p in M is the largest k for which there exists an alternating path that demotes the person $M(p)$, if any, to her last resort and then makes k people better off. If p is filled in M, this path will be a basic pressure-k improvement; if p is unfilled in M and $k > 0$, this path will be a basic Pareto improvement. Note that there always exists such a path for $k = 0$, either the path consisting of the vertex p or the path that merely demotes the occupant of p to her last resort. □

We have never explicitly said whether an alternating path in Theorem 3.3 or Definition 4.1 is allowed to use a vertex more than once; however, this is not a problem. If an alternating path P repeats a vertex, P contains a cycle, and the cycle can be removed without affecting the argument. Specifically, in Definition 4.1 and the second part of Theorem 3.3, we assumed the matching had no Pareto-improving cycle, so removing the cycle does not change the number of improvements along P. In the first part of Theorem 3.3, either the cycle or the remainder of P is a Pareto improvement.

Theorem 4.2. Let M be a matching that admits no Pareto-improving cycle. M is Pareto efficient if and only if all of its unfilled posts have pressure zero. If that is the case, the unpopularity factor of M is the greatest pressure of any of its posts.

Proof: By the remarks in Definition 4.1, if an unfilled post p_1 of M has nonzero pressure, then M has a Pareto improving path that fills p_1. Thus, for M to be Pareto efficient, all unfilled posts must have pressure zero. On the other hand, if M is Pareto inefficient, it must admit a basic Pareto improvement by Theorem 3.3. However, we assumed M admits no Pareto-improving cycle, so this improvement must be a path; suppose it ends at a (necessarily unfilled) post p_1. According to Definition 4.1, the pressure of p_1 is at least the number of people improved by the path, so p_1 has nonzero pressure. This completes the proof of the first part.

For the second part, let f be the unpopularity factor of M. By Theorem 3.3, there exists a basic pressure-f improving path over M, and this path contributes to the pressure of the post from which it demotes a person. Thus, the greatest pressure is at least f. If there were a pressure k greater than f, it would be nonzero and thus on a filled post p_1. However, that pressure could only result from a basic pressure-k improvement, and Theorem 3.3 tells us that no basic pressure-k improvement exists for $k > f$. □

A simple algorithm based on the Bellman-Ford shortest-path algorithm will find the pressures of any matching.

Algorithm 4.3. Finds the pressure of each post in a matching M or reports that M admits a Pareto-improving cycle. The algorithm runs in $O(pe)$ time, where p is the number of posts and e is the total number of entries of the preference lists.

Method: Construct a graph G whose vertices are the posts of M and whose edges are as follows:

- There is an edge of weight -1 from p_1 to p_2 if $M(p_1)$ strictly prefers p_2 to p_1.
- There is an edge of weight 0 from p_1 to p_2 if $M(p_1)$ is indifferent between p_1 and p_2.
- There is no edge from p_1 to p_2 if $M(p_1)$ strictly prefers p_1 to p_2, $M(p_1)$'s preference list omits p_2, or p_1 is unmatched in M (so there is no $M(p_1)$).

Run the Bellman-Ford shortest-path algorithm on G. Register all vertices as sources so that the algorithm finds the shortest path beginning at any vertex; alternatively, create an artificial source vertex s and add an edge of weight 0 from s to each other vertex. If the Bellman-Ford algorithm reports that G has a negative cycle, announce that M admits a Pareto-improving cycle. Otherwise, the pressure of a post p_1 is the negative of the shortest distance to p_1.

Correctness: Each edge in G represents a possible reassignment in M that does not make the reassigned person worse off. Specifically, an edge $e = (p_1, p_2)$ in G represents the movement of the person $M(p_1)$ from post p_1 to post p_2. e has weight -1 if $M(p_1)$ wants the reassignment and 0 if $M(p_1)$ is indifferent to the reassignment. Thus, a path P_G in G represents an alternating path P_M that makes no one worse off and is applicable to M after we first vacate the ending post if necessary. P_M is exactly the kind of path considered in Definition 4.1. If P_G has length $-k$, it contains k edges of weight -1 (each representing a reassignment that promotes one person),

so P_M improves k people. Thus, finding shortest paths in G corresponds in a rather direct manner to finding pressures in M.

With this background, we can prove the correctness of the algorithm:

- A negative cycle in G is just a cycle of posts that contains at least one edge of weight -1. Such cycles in G correspond to Pareto-improving cycles in M and vice versa. Thus, the Bellman-Ford algorithm announces a negative cycle if and only if M has a Pareto-improving cycle.
- Otherwise, let p_1 be a vertex of G, and suppose the Bellman-Ford algorithm determines that the shortest distance to p_1 (from any vertex) is $-k$. That means k is the maximum possible number of edges of weight -1 on a path from any vertex to p_1 in G. A path to p_1 in G containing s edges of weight -1 corresponds to an alternating path ending at p_1 in M that improves s people after possibly demoting $M(p_1)$ to her last resort, and vice versa. Thus, k is also the maximum number of people who can improve in a path to p_1, i.e., the pressure of p_1.

Running time: Each edge (p_1, p_2) of G corresponds to a unique preference-list entry $(M(p_1), p_2)$, so the number of edges does not exceed the total number of preference-list entries. The Bellman-Ford algorithm takes $O(VE)$ time, where $V = p$ and $E \leq e$, and the additional work in Algorithm 4.3 is fast. □

Algorithm 4.4. Determines the unpopularity factor of a matching M in $O(pe)$ time.

Method: Use Algorithm 4.3 to find the pressures of M. If Algorithm 4.3 reports that M admits a Pareto-improving cycle or gives a nonzero pressure for a post unfilled in M, announce that M has unpopularity factor ∞. Otherwise, take the maximum of the pressures as the unpopularity factor of M.

Correctness: A consequence of Theorem 4.2.

Running time: Algorithm 4.3 runs in $O(pe)$ time, and the additional work in this algorithm is faster. □

The author has implemented Algorithms 4.3 and 4.4 in Java; the program is available upon request.

One may obtain faster versions of these algorithms that run in only $O(\sqrt{p}e)$ time by computing shortest paths using Goldberg's algorithm [5] instead of the Bellman-Ford algorithm. Furthermore, if M is a matching of an instance whose preference lists contain no ties, G contains only edges of weight -1, so an $O(e)$ topological sort of the vertices of G suffices to find the pressures of M.

5 Least-unpopularity-factor matching is NP-hard

In this section, we use a reduction from 3-satisfiability to prove that least-unpopularity-factor matching is in general NP-hard. Specifically, we present a construction that converts an instance I_S of 3-satisfiability to a polynomial-size instance I_M of least-unpopularity-factor matching. We then show that any matching for I_M of unpopularity factor at most 2 can be converted to a set of truth values satisfying I_S and vice versa. That means that the least unpopularity factor of I_M is at most 2 exactly when I_S is satisfiable, and an algorithm to find a matching of least unpopularity factor can certainly tell us what that factor is for I_M.

Abraham et al. [1] analyze popular matching instances with no ties in the preference lists separately from the general case of ties. In previous sections of this paper, there has been no reason to make this distinction. However, the construction to follow will always generate matching instances with no ties; that will show that finding least-unpopularity-factor matchings is NP-hard even when preference lists are guaranteed not to have ties.

5.1 Gadgets

We will present a reduction from 3-satisfiability that, like most, is based on *gadgets* that represent pieces of I_S and are built from elements of least-unpopularity-factor matching. Intuitively, I_S is broken down into simpler constraints, each of which can be enforced by a single gadget of I_M. If I_M is matched in a way that violates a constraint, the corresponding gadget produces a pressure of at least 3; otherwise the pressures inside the gadget are at most 2. Thus, a matching of I_M has unpopularity factor at most 2 if and only if it satisfies all the constraints.

When we link gadgets together, we must prevent pressures from being transmitted from one gadget to another so that we can analyze the pressures inside each gadget separately. To this end, we will design gadgets

as follows. Each gadget will have several *internal* people and posts. The preference list of a person internal to a gadget will contain only posts internal to that gadget and, of course, his last resort. Each gadget will also have one or more *linking people*. A linking person's preference list consists of posts internal to the gadget and perhaps one *linking post* not internal to any gadget; the linking post, if any, must be strictly preferred to all other posts on the list. A linking post is on the preference lists of at most two people.

Each linking post will be occupied by one linking person; the other linking person who likes that post must instead occupy a post internal to his own gadget. The occupant of a linking post never pressures other posts, so we need only consider the pressures inside each gadget and those transmitted to the gadget's linking posts by its linking people matched inside it, if any.

5.2 Boxes

One gadget we will use is called a *box*. It has four internal posts, three internal people (i_1, i_2, and i_3), and three linking people (w, n_1, and n_2). Its structure is shown below, including the linking posts that are shared with other gadgets.

$$\begin{array}{c} \\ i_1 \\ i_2 \\ i_3 \\ w \\ n_1 \\ n_2 \end{array} \begin{array}{c} \begin{array}{ccccccc} x & y & z & u & l_w & l_{n1} & l_{n2} \end{array} \\ \begin{pmatrix} 1 & 2 & 3 & 4 & - & - & - \\ 1 & 2 & 3 & 4 & - & - & - \\ 1 & 2 & 3 & 4 & - & - & - \\ 2 & 3 & 5 & 4 & 1 & - & - \\ - & - & - & 2 & - & 1 & - \\ - & - & - & 2 & - & - & 1 \end{pmatrix} \end{array}$$

w is known as the *wide* person, and n_1 and n_2 are the *narrow* people. The internal posts of the box can accomodate either the wide person or both narrow people without generating a pressure above 2 on any of the posts shown.

Below is the matching of the box that accommodates the wide person. Superscripts indicate pressures. If a linking person from this box occupies a linking post, then the status of the other person who likes the post determines the post's pressure, so we need not consider that pressure now.

$$\begin{array}{c} \\ i_1 \\ i_2 \\ i_3 \\ w \\ n_1 \\ n_2 \end{array} \begin{array}{c} \begin{array}{ccccccc} x^2 & y^1 & z^0 & u^0 & l_w^1 & l_{n1} & l_{n2} \end{array} \\ \begin{pmatrix} \underline{\mathbf{1}} & 2 & 3 & 4 & - & - & - \\ 1 & \underline{\mathbf{2}} & 3 & 4 & - & - & - \\ 1 & 2 & \underline{\mathbf{3}} & 4 & - & - & - \\ 2 & 3 & 5 & \underline{\mathbf{4}} & 1 & - & - \\ - & - & - & 2 & - & \underline{\mathbf{1}} & - \\ - & - & - & 2 & - & - & \underline{\mathbf{1}} \end{pmatrix} \end{array}$$

In similar fashion, here are the matchings that accommodate one narrow person and both narrow people, respectively. In the latter matching, n_2 is assigned to his last resort.

$$\begin{array}{c} \\ i_1 \\ i_2 \\ i_3 \\ w \\ n_1 \\ n_2 \end{array} \begin{array}{c} \begin{array}{ccccccc} x^2 & y^1 & z^0 & u^0 & l_w & l_{n1}^1 & l_{n2} \end{array} \\ \begin{pmatrix} \underline{\mathbf{1}} & 2 & 3 & 4 & - & - & - \\ 1 & \underline{\mathbf{2}} & 3 & 4 & - & - & - \\ 1 & 2 & \underline{\mathbf{3}} & 4 & - & - & - \\ 2 & 3 & 5 & 4 & \underline{\mathbf{1}} & - & - \\ - & - & - & \underline{\mathbf{2}} & - & 1 & - \\ - & - & - & 2 & - & - & \underline{\mathbf{1}} \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ i_1 \\ i_2 \\ i_3 \\ w \\ n_1 \\ n_2 \end{array} \begin{array}{c} \begin{array}{ccccccc} x^2 & y^1 & z^0 & u^1 & l_w & l_{n1}^2 & l_{n2}^1 \end{array} \\ \begin{pmatrix} \underline{\mathbf{1}} & 2 & 3 & 4 & - & - & - \\ 1 & \underline{\mathbf{2}} & 3 & 4 & - & - & - \\ 1 & 2 & \underline{\mathbf{3}} & 4 & - & - & - \\ 2 & 3 & 5 & 4 & \underline{\mathbf{1}} & - & - \\ - & - & - & \underline{\mathbf{2}} & - & 1 & - \\ - & - & - & 2 & - & - & 1 \end{pmatrix} \end{array}$$

However, we claim that if enough linking posts are filled by linking people from other gadgets that the wide person and at least one narrow person are forced into the box, the box generates a pressure of 3. Without loss of generality, suppose n_1 is matched inside the box; n_2 may be matched inside or outside.

Theorem 5.1. If M is a Pareto efficient matching of I_M in which w and n_1 are matched inside the box, then the unpopularity factor of M is at least 3.

Proof: Suppose one of x, y, z, or u were unfilled in M. Since they are the only possible non-last-resort posts for i_1, i_2, i_3, and w, at least one of these four people would be matched to his last resort by the Pigeonhole

Principle. He could be promoted to the unfilled post, so M would be Pareto inefficient, a contradiction. That means x, y, z, and u are all filled in M.

Since we have assumed that w and n_1 are both matched inside the box, the only possible non-last-resort posts for i_1, i_2, i_3, w and n_1 are x, y, z, and u. By the Pigeonhole Principle, one of these five people is matched to his last resort. We have two cases:

- *n_1 is matched to his last resort:* n_1 pressures u. We showed earlier that u is filled, and it must be filled by i_1, i_2, i_3, or w. All of these people prefer y to u, so the occupant of u pressures y. Thus the pressure of y is at least 2.
- *One of i_1, i_2, i_3, w is matched to his last resort:* That person pressures z. As before, z is filled and its occupant pressures y, so the pressure of y is at least 2.

The occupant of y pressures x, so the pressure of x is at least 3. By Theorem 4.2, the unpopularity factor of M is at least 3, as desired. □

5.3 Chaining boxes

A box is a "two-for-one" device: either its one wide linking post or its two narrow linking posts can be occupied by people belonging to other gadgets without generating an unacceptable pressure. If we identify the wide linking post of one box and a narrow linking post of another box, we get a three-for-one device, and so forth.

Let $x_1, \ldots, x_n$ be the variables of I_S. For each i, suppose x_i appears as a factor u times in all the clauses of I_S, and suppose $\neg x_i$ appears v times. If $u = 0$ or $v = 0$, we can fix $x_i =$ *false* or $x_i =$ *true*, respectively, and discard it. Otherwise, chain $u - 1$ boxes to get a u-for-one device and chain $v - 1$ boxes to get a v-for-one device. Now identify the wide posts of these two devices to get a u-for-v device. Each linking post of the resulting device represents an occurrence of x_i or $\neg x_i$.

Figure 3 shows what the device for a variable x might look like if x and $\neg x$ each appear four times in I_S. Boxes represent boxes, circles represent linking posts, and lines represent linking people. Internal people and posts are not shown.

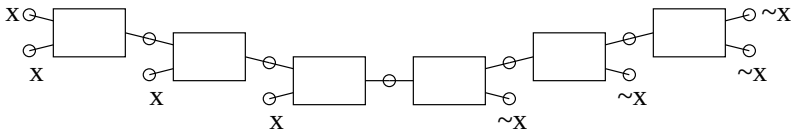

Figure 3: A chain of boxes representing one variable.

5.4 Pools

For each clause of I_S, generate a copy of the simpler gadget below, called a *pool*. x, y, and z are internal posts; a pool has no internal people. f_1, f_2, and f_3 are linking people, and l_{f1}, l_{f2}, and l_{f3} are their respective linking posts. One linking post corresponds to each variable reference in the clause.

$$\begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{cccccc} x & y & z & l_{f1} & l_{f2} & l_{f3} \end{array} \\ \begin{pmatrix} 2 & 3 & 4 & 1 & - & - \\ 2 & 3 & 4 & - & 1 & - \\ 2 & 3 & 4 & - & - & 1 \end{pmatrix} \end{array}$$

The pool can accomodate up to two of its linking people with pressures not exceeding 2, but if all three linking people are matched inside the pool, a pressure of 3 will be generated on one of the linking posts:

$$\begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{cccccc} x^0 & y^0 & z^0 & l_{f1} & l_{f2} & l_{f3} \end{array} \\ \begin{pmatrix} 2 & 3 & 4 & \underline{\mathbf{1}} & - & - \\ 2 & 3 & 4 & - & \underline{\mathbf{1}} & - \\ 2 & 3 & 4 & - & - & \underline{\mathbf{1}} \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{cccccc} x^0 & y^0 & z^0 & l^1_{f1} & l_{f2} & l_{f3} \end{array} \\ \begin{pmatrix} \underline{\mathbf{2}} & 3 & 4 & 1 & - & - \\ 2 & 3 & 4 & - & \underline{\mathbf{1}} & - \\ 2 & 3 & 4 & - & - & \underline{\mathbf{1}} \end{pmatrix} \end{array}$$

$$\begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} x^1 & y^0 & z^0 & l^2_{f1} & l^1_{f2} & l_{f3} \end{array} \\ \begin{pmatrix} \underline{\mathbf{2}} & 3 & 4 & 1 & - & - \\ 2 & \underline{\mathbf{3}} & 4 & - & 1 & - \\ 2 & 3 & 4 & - & - & \underline{\mathbf{1}} \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} x^2 & y^1 & z^0 & l^3_{f1} & l^2_{f2} & l^1_{f3} \end{array} \\ \begin{pmatrix} \underline{\mathbf{2}} & 3 & 4 & 1 & - & - \\ 2 & \underline{\mathbf{3}} & 4 & - & 1 & - \\ 2 & 3 & \underline{\mathbf{4}} & - & - & 1 \end{pmatrix} \end{array}$$

5.5 Putting it all together

To create I_M, only one more step is needed. For each variable reference x_i or $\neg x_i$ in I_S, identify a linking post from the clause's pool with a linking post representing x_i or $\neg x_i$ (as appropriate) from the ith chain. To avoid a pressure of 3, at least one of the three people belonging to each pool must be matched outside the pool, subject to the constraints imposed by the boxes. With this rather suggestive condition in mind, we can prove the correctness of the conversion.

Algorithm 5.2. Converts an instance I_S of 3-satisfiability to an instance I_M of least-unpopularity-factor matching in such a way that I_S is satisfiable if and only if the least unpopularity factor of I_M is at most 2. Takes time polynomial in the size of I_S and produces an I_M whose size is polynomial in that of I_S.

Method: As described in the text.

Correctness: We will show that, from any solution to I_S, we can construct a "solution" to I_M (i.e., a matching of unpopularity factor at most 2), and vice versa.

I_S solution $\mapsto$ I_M solution: Let $x_1, \ldots, x_n$ be the variables of I_S, and let $t_1, \ldots, t_n$ be one set of truth values that satisfies I_S. Construct a matching M for I_M as follows. For each i, if $t_i =$ *true*, then match the ith chain as follows. On the x_i side, match all wide people to their linking posts and all narrow people inside their boxes; in particular, fill the central linking post of the chain with the wide person from the first box to the x_i side. On the $\neg x_i$ side, match all wide people inside their boxes and all narrow people to linking posts. By the discussion in sections 5.2 and 5.3, this can be accomplished without a pressure exceeding 2.

This procedure leaves free the linking posts that represent references to x_i; fill those with linking people from pools. If instead $t_i =$ *false*, do it the other way around, leaving free the linking posts that represent references to $\neg x_i$, and fill those with linking people from pools. After performing this procedure for each i, match the remaining pool linking people inside their respective pools.

Since the t_i satisfy I_S, at least one reference in each clause evaluates to *true* given the t_i. The linking post corresponding to that reference is not filled by a person from the ith chain, so it is filled by a person from the clause's pool. Thus, at most two of the linking people of each pool need be matched to its internal posts, and we showed in section 5.4 that this can be done without generating a pressure greater than 2. By Theorem 4.2, M has unpopularity factor at most 2, as desired.

I_M solution $\mapsto$ I_S solution: Let $x_1, \ldots, x_n$ be the variables of I_S, and let M be a matching of I_M of unpopularity factor at most 2. For each i, consider the central linking post of the ith chain. Exactly two people, a wide person from the x_i side of the chain and a wide person from the $\neg x_i$ side, list the central post in their preferences, and it is the favorite post of both. Since M is Pareto efficient, one of these people must fill the central post. If the x_i person fills it, set $x_i =$ *true*, and if the $\neg x_i$ person fills it, set $x_i =$ *false*. We claim that the resulting truth values satisfy I_S.

Suppose we have set $x_i =$ *true* because M assigns the x_i wide person to the central post of the ith chain. Clearly M must match the $\neg x_i$ wide person inside the first box on the $\neg x_i$ side of center. According to section 5.2, both of that box's narrow people must be matched to linking posts to avoid generating a pressure of 3. One of those people fills the linking post of the wide person of the second box, who therefore must be matched inside the second box. Continuing in this fashion, we find that on the $\neg x_i$ side, all wide people are matched inside their boxes and all narrow people are matched to linking posts. In particular, all of the posts representing references to $\neg x_i$ are filled with linking people from the chain. That means the linking people from the pools of clauses referring to $\neg x_i$ must be matched inside those pools. Similarly, if we have set $x_i =$ *false*, then the linking people from the pools of clauses referring to x_i must be matched inside those pools.

However, we showed in section 5.2 that M must match at least one of the three linking people from each pool to a linking post to avoid generating a pressure of 3; call that person p. If p corresponds to a reference x_j, we must have set $x_j =$ *true*, because if we had set $x_j =$ *false*, p's post would have been occupied by someone from the jth chain. Similarly, if p corresponds to a reference $\neg x_j$, we must have set $x_j =$ *false*. Thus, at least

one reference from the pool (i.e., clause) evaluates to *true* given our truth values. This argument applies to every pool/clause, so the truth values we assigned to the x_i satisfy I_S.

Running time: At most a constant number of people and a constant number of posts are generated from each variable or variable reference, so the size of I_M is in fact linear in that of I_S. It should then be clear that the conversion runs in polynomial time. □

Corollary 5.3. The problem of deciding whether the least unpopularity factor of an instance is ≤ 2 or ≥ 3 is NP-complete. □

Corollary 5.4. Finding least-unpopularity-factor matchings in general is NP-hard. □

The author's Java implementation includes a routine to transform 3-satisfiability instances to matching instances.

6 Conclusion

Least-unpopularity-factor matching is a new approach to assignment problems with one-sided preferences; it improves upon popular matching by designating one or more matchings to be "as popular as possible" even if no popular matching exists. We have shown that the problem of finding a matching of least unpopularity factor is NP-hard, which means that no algorithm is likely to be discovered that efficiently finds such a matching for large numbers of people and positions. This result represents a small but significant step toward the goal of assigning people to posts automatically by computer.

Future work could explore least-unpopularity-factor matching further by finding an approximation algorithm (which would compute a matching whose unpopularity is within a constant factor of the least possible) or proving an inapproximability bound. In search of an approximation algorithm, the author implemented three conjectured algorithms. All three produce close-to-optimal matchings for most small instances, but the author could not prove an approximation factor for any of them.

However, the central goal of future work should be to find a better assignment rule that has all the necessary properties to be generally useful. *Least-unpopularity-margin matching* might be such a criterion; it is a second generalization of popular matching obtained by replacing u/v with $u - v$ in Definition 3.1. The difference seems small, but it changes unpopularity from an essentially local property (as we saw in Theorem 3.3) to an essentially global one. That makes the existence of an approximation algorithm much more likely; indeed, it may be possible to find exact least-unpopularity-margin matchings in polynomial time. If that is the case, humans need never construct matchings by hand again.

References

[1] David J. Abraham, Robert W. Irving, Telikepalli Kavitha, and Kurt Mehlhorn. *Popular matchings.* Proceedings of SODA 2005: the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, 424–432. Available at `http://www.mpii.mpg.de/~mehlhorn/ftp/PopularMatchings.ps` as of 9/26/2006.

[2] David J. Abraham and Telikepalli Kavitha. *Dynamic matching markets and voting paths.* Proceedings of SWAT 2006: the 10th Scandinavian Workshop on Algorithm Theory, 65–76. Available at `http://www.cs.cmu.edu/~dabraham/papers/voting_paths-swat.pdf` as of 9/26/2006.

[3] David Gale and Lloyd S. Shapley. *College admissions and the stability of marriage.* American Mathematical Monthly, 16:9–15, 1962.

[4] Peter Gärdenfors. *Match Making: assignments based on bilateral preferences.* Behavioural Sciences, 20:166–173, 1975.

[5] Andrew V. Goldberg. *Scaling algorithms for the shortest paths problem.* Proceedings of SODA 1993: the 4th Annual ACM-SIAM Symposium on Discrete Algorithms, 222–231.

[6] Robert W. Irving, Telikepalli Kavitha, Kurt Mehlhorn, Dimitrios Michail, and Katarzyna Paluch. *Rank-maximal matchings.* Proceedings of SODA 2004: the 15th Annual ACM-SIAM Symposium on Discrete Algorithms, 68–75. Available at `http://www.mpi-sb.mpg.de/~michail/papers/RankMaximalMatchings-journal.ps.gz` as of 9/26/2006.

[7] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Chapter 11, "Weighted Matching." Prentice-Hall, 1982.

[8] Eric Price. Personal communication, August 2005.

[9] Alvin E. Roth. *The evolution of the labor market for medical interns and residents: A case study in game theory.* Journal of Political Economy, 92:991–1016, 1984.