The least-unpopularity-factor and least-unpopularity-margin criteria for matching problems with one-sided preferences

Richard Matthew McCutchen*

Abstract

Given a set of people, a set of positions available to them (e.g., houses or jobs), and each person's preference ordering of the positions, how shall we assign the people to the positions? Since different matchings inevitably favor different people, it is not clear which matching(s) should be considered best overall; many different optimality criteria are possible. One is popularity: a matching is popular if no other matching beats it in a majority vote of the people. Popularity has a property that helps it resist manipulation and popular matchings can be found quickly when they exist, but for some sets of preferences, no matching is popular. In this paper, we present the least-unpopularity-factor and least-unpopularity-margin criteria, two generalizations of popularity that preserve the resistance to manipulation but give an optimal matching for every set of preferences. Under each of these two generalizations, we show that the "badness" of a given matching can be calculated efficiently but it is NP-hard to find an optimal matching. This work represents a step toward the goal of a computer program to automatically solve real-world matching problems according to a good optimality criterion.

1 Introduction

One of the most common administrative tasks that many organizations perform is assigning people to positions of some kind 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 choosing a matching of people to positions that gives fair consideration to everyone's preferences. In this paper, we consider the problem in which the people express preferences for the positions, but not vice versa: the preferences are *one-sided*. An *instance* of this problem consists of a set of people, a set of positions, and a *preference list* for each person giving his/her preference ordering of the subset of the positions that he/she would be willing to occupy; these orderings may contain ties. The problem is to find the matching that is best overall in light of the preferences. In a valid matching, each person is either matched to a position on his/her preference list or left without a position. For convenience, we think of each

*Department of Computer Science, University of Maryland, College Park, MD 20742. Email: rmccutch@umd.edu .

person p as having a *last resort* position $LR(p)$ available only to p as p's last choice; assigning a person to his/her last resort represents leaving him/her without a position.

What makes this problem interesting is that it is not clear what it means for a matching to be "best". Since it is almost never possible to give everyone his/her first choice simultaneously, any matching will inevitably make trade-offs, pleasing some people at the expense of others. If we can change a matching to make someone better off without making anyone else worse off (a *Pareto improvement*), of course we should do so, but there are still many *Pareto efficient* matchings that admit no such improvement, and each makes a different set of trade-offs. To decide among these, we need an *optimality criterion* that, for each instance, designates one or more matchings as optimal. Then, to apply the criterion in practice, we need an efficient algorithm to compute an optimal matching for a given instance. Ideally, our criterion should be "fair" in some sense and should resist attempts by the people to obtain better positions by lying about their preferences.

Many different optimality criteria have been proposed, studied, and used. Some optimality criteria are based on minimizing various functions of the numerical preference ranks that the people give to the positions they receive, but unfortunately, such criteria tend to be easy to manipulate. For example, at one point MIT assigned incoming students to residence halls by minimizing the sum of the cubes of the numerical ranks, and a student could improve his/her chances of being assigned to his/her true first-choice residence hall by inserting other highly desirable residence halls near the top of his/her preference list [8].

One criterion that does not use numerical ranks and is therefore less susceptible to this kind of manipulation is *popularity*. A matching M is *popular* if there does not exist another matching N that beats it by majority vote, where each person votes for the matching that gives him/her the position he/she likes better or abstains if he/she likes the two positions equally well. For example, consider the instance I and matchings M and N shown below. (In this paper, we will display instances as tables that give the preference rank number (or $-$ if forbidden) for each person (row) and position (column); last resort positions are never shown. We will display a matching as a set of parenthesized rank numbers. Here, I contains three people A, B, C and three positions w, x, y. Person B likes position w better than position x and is unwilling to occupy position y. In matching M, he occupies position x.)

$$I = \begin{array}{c} \\ A \\ B \\ C \end{array}\!\!\begin{array}{c} \begin{array}{ccc} w & x & y \end{array} \\ \begin{pmatrix} 1 & - & 2 \\ 1 & 2 & - \\ - & 1 & 2 \end{pmatrix} \end{array}, \qquad M = \begin{array}{c} \\ A \\ B \\ C \end{array}\!\!\begin{array}{c} \begin{array}{ccc} w & x & y \end{array} \\ \begin{pmatrix} (\mathbf{1}) & - & 2 \\ 1 & (\mathbf{2}) & - \\ - & 1 & (\mathbf{2}) \end{pmatrix} \end{array}, \qquad N = \begin{array}{c} \\ A \\ B \\ C \end{array}\!\!\begin{array}{c} \begin{array}{ccc} w & x & y \end{array} \\ \begin{pmatrix} 1 & - & (\mathbf{2}) \\ (\mathbf{1}) & 2 & - \\ - & (\mathbf{1}) & 2 \end{pmatrix} \end{array}$$

M is not popular because N beats it by majority vote (A and C outvote B in favor of N), while one can easily verify that N is popular. Abraham et al. [1] gave an efficient algorithm to compute a popular matching for a given instance when one exists. Unfortunately, some instances, such I_2, have no popular matching because of nontransitivity in the voting:

$$I_2 = \begin{array}{c} \\ A \\ B \\ C \end{array}\!\!\begin{array}{c} \begin{array}{ccc} w & x & y \end{array} \\ \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix} \end{array}$$

Given any matching, we can obtain a matching that beats it by a vote of 2 to 1 by promoting the occupant of y to x and the occupant of x to w and demoting the occupant of w to y.

In general, when no matching meets the standard of popularity, we would still like to choose a merely "poor" matching over a "terrible" one. To this end, we will propose two numerical

measures of a matching's "badness": its *unpopularity factor* and its *unpopularity margin*. The *least-unpopularity-factor* and *least-unpopularity-margin* optimality criteria, respectively, are based on the minimization of these two measures. We will show that a given matching's unpopularity factor and unpopularity margin can be calculated efficiently using algorithms based on shortest paths and minimum-cost flow, respectively. However, surprisingly, it is possible to encode the structure of a 3-satisfiability instance in the people's preferences so as to prove that finding an optimal matching under either criterion is NP-hard; we will present this rather unusual reduction in detail.

The inablity to find optimal matchings severely limits the immediate practical applicability of these two criteria. Nevertheless, by ruling them out, my work makes a step toward the goal of finding a criterion that has all the desired properties. Such a criterion would make it possible for organizations to solve their matching problems easily, fairly, and objectively by computer, which would constitute a significant advance over the currently widespread technique of eyeballing all the preferences and manually constructing a matching one person at a time.

2 Related work

The notion of popularity was first introduced by Gärdenfors [4] in the context of two-sided preferences. More recently, Abraham et al. [1] discussed it for one-sided preferences and gave the first polynomial-time algorithm to find popular matchings. On an instance with n people and a total of m preference-list entries, the algorithm runs in $O(\sqrt{n}m)$ if there are ties and $O(n + m)$ if not. Abraham and Kavitha [2] suggested that, when no popular matching exists, a matching could be chosen 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-maximality is another criterion. 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 positions they respectively ranked ith. 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 in the result (e.g., 3 for a third choice).

If each person gives numerical values to the positions instead of ordering them, the most natural rule, known as *weighted matching*, is to maximize the total value of all pairs. In instances with two-sided preferences (we think of people as applicants and positions as recruiters), the most commonly used rule is *stable marriage*. A matching is *stable* if no currently unpaired applicant and recruiter both prefer each other to their current partners. Gale and Shapley [3] proposed the rule, proved that a stable matching always exists, and gave an algorithm to find one.

3 The unpopularity factor

We will begin by defining the unpopularity factor of a matching.

Definition 3.1. If M and N are two matchings for the same instance, N *dominates* M *by a factor of* u/v, 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.

Definition 3.2. The *unpopularity factor* of a matching M is the maximum factor by which it is dominated by any other matching (ignoring matchings that give $u = v = 0$).

Note that a matching N dominates a matching M by a factor of ∞ if and only if N is a Pareto improvement over M; thus M has a finite unpopularity factor if and only if it is Pareto efficient.

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

The *least unpopularity factor* of an instance is the minimum unpopularity factor of all of its matchings, and the matching(s) that achieve this minimum are considered optimal. Thus, different instances have different numbers of optimal matchings, but every instance has at least one.

Let us revisit the examples from the Introduction. M has unpopularity factor 2 because N dominates it by a factor of $2/1 = 2$; it is clear by inspection that no other matching dominates M by a greater factor. However, N 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 a dominance factor of $1/1 = 1$. In I_2, the six matchings that fill all three positions 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 optimal for I_2.

3.1 Differences between matchings: reassignments, paths and cycles

To determine the unpopularity factor of a matching directly from the definition, we would have to consider all possible alternative matchings and calculate the factor by which each dominates M, which would take exponential time. Fortunately, there is an efficient way to calculate the unpopularity factor using the concept of *pressures*, which we will develop next.

We can think of an instance as a bipartite graph whose vertex sets are the people and the positions and whose edges are the preference-list entries. A matching of the instance is then just a matching of the graph that uses all the people (because every person has a position, though it might just be her last resort).

In what ways can two matchings M and N differ? Their symmetric difference, which we will denote $M \oplus N$, consists of vertex-disjoint paths and/or cycles that are alternating in the sense that their edges come alternately from M and N. Furthermore, the alternating paths stop at positions, because if a path stopped at a person, she would lack a position in one matching, which is not allowed. We can think of an alternating path as a sequence of reassignments: one person moves to a different position, ejecting its original occupant; the occupant takes another position, ejecting *its* occupant; and so forth until someone takes a previously unoccupied position. Each reassignment may constitute a *promotion* or *demotion* of the reassigned person according to his/her preferences. An alternating cycle is similar except that all reassignments must "happen at once".

Conversely, we say that a path or cycle X is *applicable* to a matching M if $M \oplus X$ is a valid matching. If so, *applying* X to M gives $M \oplus X$; X represents the change from M to $M \oplus X$.

With this intuition, we can define pressures.

3.2 Pressures

Definition 3.3. Let M be a Pareto efficient matching. The *pressure* of a filled position p in M is the largest k for which there exists an alternating path applicable to M that promotes k people without demoting anyone and then ends with the demotion of the occupant of p to her last resort. Note that the demotion by itself constitutes such a path for $k = 0$. (The term "pressure" comes from the idea of k people stacked up behind p, wishing that its occupant will leave so they can all become better off.)

Theorem 3.4. *The unpopularity factor of a Pareto efficient matching M is the greatest pressure of any of its filled positions.*

Proof. Recall that the unpopularity factor of M is the greatest pressure by which any other matching dominates M. To establish that the two maxima are equal, we'll show that each is at least as

great as the other, i.e., (a) if M has a position p of pressure k, then there exists a matching N that dominates M by a factor of at least k and (b) if a matching N dominates M by a factor of f, then M has a position of pressure at least $\lceil f \rceil$.

(a): Simply let N be the result of applying the path that determines the pressure of p to M. The path makes k people better off and one person worse off, so N dominates M by a factor of k.

(b): Let u and v be the total number of people better and worse off in N than in M, so that $f = u/v$. We begin by modifying N so that everyone who is worse off in N than in M is demoted all the way to his last resort in N; this does not change u or v. Decompose $M \oplus N$ into a collection of paths and cycles $X_1, \ldots, X_c$, discarding those that neither promote nor demote anyone. Each X_i must demote at least one person so that it is not a Pareto improvement over M. That person is demoted to his last resort, and since there is no one else to leave the last resort, the demotion must be the last reassignment in X_i; thus X_i is a path (not a cycle). A path has only one last reassignment, so each X_i demotes *exactly* one person to his last resort, and $c = v$.

For each i, let u_i be the number of people promoted by X_i; of course, $\sum_i u_i = u$. Choose an i such that $u_i \geq \lceil u/v \rceil = \lceil f \rceil$; by the Pigeonhole Principle, one must exist. Let p be the position whose occupant is demoted by X_i. Observe that X_i has exactly the form considered in Definition 3.3 for the pressure of p, and it makes at least $\lceil f \rceil$ people better off; thus the pressure of p is at least $\lceil f \rceil$. □

Corollary 3.5. *The unpopularity factor of a matching, if finite, is an integer.*

3.3 Computing the unpopularity factor

We can find the pressures of a matching using the following simple algorithm based on Goldberg's shortest-path algorithm [5]:

Algorithm 3.6. *Determines whether a matching M is Pareto efficient and, if so, finds the pressure of each filled position in M. Runs in $O(m\sqrt{n})$ time, where n is the number of positions and m is the total number of entries of the preference lists.*

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

- There is an edge of length -1 from p_1 to p_2 if p_1 is filled by a person who would strictly prefer to have p_2.
- There is an edge of length 0 from p_1 to p_2 if p_1 is filled by a person who is indifferent to having p_2.
- There is no edge from p_1 to p_2 if p_1 is empty *or* its occupant strictly prefers p_1 over p_2 or is unwilling to occupy p_2.

Run Goldberg's algorithm on G, using all positions as sources so that the algorithm finds the shortest path arriving at each position. If Goldberg's algorithm finds a negative cycle in G or a negative-length path arriving at an unfilled position, announce that M is Pareto inefficient. Otherwise, the pressure of each position is the negative of the length of the shortest path arriving at it. □

Correctness. Each edge $e = (p_1, p_2)$ in G represents the reassignment of the occupant of p_1 to p_2; the edges of G correspond exactly to the reassignments that are not demotions. Furthermore, e has weight -1 if and only if the reassignment is a promotion, so the length of a path or cycle in G is the negative of the number of people it promotes.

In light of this, a negative cycle in G represents a Pareto-improving cycle applicable to M, and a negative-length path to an unfilled position represents a Pareto-improving path in M; thus the algorithm's determination of whether M is Pareto efficient is correct. If M is Pareto efficient, paths in G arriving at a position p represent paths applicable to M of the form considered in Definition 3.3 with the negative of a path's length corresponding to k in that definition. Thus, the negative of the shortest length of a path arriving at p gives the pressure of p, as desired. □

Running time. Each edge (p_1, p_2) of G corresponds to a different preference-list entry in M, so the number of edges does not exceed the total number of preference-list entries. Since edge lengths are at least -1, Goldberg's algorithm runs in $O(m\sqrt{n})$ time, and the additional overhead in this algorithm is less than that. □

To find the unpopularity factor of a matching, we use Algorithm 3.6 to find the pressures and then simply take the highest pressure.

4 The unpopularity margin

The unpopularity margin of a matching is defined the same way as the unpopularity factor, except we subtract the numbers of votes instead of dividing them:

Definition 4.1. If M and N are two matchings for the same instance, N *dominates* M *by a margin of* $u - v$, 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.

Definition 4.2. The *unpopularity margin* of a matching M is the maximum margin by which it is dominated by any other matching.

Note that the unpopularity margin of a matching M is an integer, and it is nonnegative because M dominates itself by a margin of zero. Just as we can express calculating the unpopularity factor as a shortest-path problem, we can express calculating the unpopularity margin as an min-cost max-flow problem. Since we use integer edge capacities, we assume that edge flows are also integers.

Algorithm 4.3. *Finds the unpopularity margin of a matching M in $O((u+1)m\sqrt{r+s})$ time, where r is the number of people, s is the number of positions, m is the total number of entries of the preference lists, and u is the unpopularity margin.*

Method and correctness. Construct a flow graph G having as vertices a source, a sink, and the people and positions of M. Add an edge of unit capacity and zero cost from the source to each person and from each position to the sink. For each preference-list entry submitted by a person A for a position p, add a unit-capacity edge from A to p whose cost is -1, 0 or 1 as A likes p better than, the same as, or worse than her position in M.

The flow that sends one unit from the source to each person to his/her last resort to the sink transports r units of flow and saturates all edges leaving the source, so it is a max-flow; thus every max-flow transports r units of flow. To do so, a max-flow must put one unit of flow through each person, and those units must reach the sink via different positions since the capacity from each position to the sink is limited to 1. Thus, the max-flows of G correspond exactly to the possible matchings of the instance.

Furthermore, it should be clear that the cost of a max-flow is the negative of the margin by which it dominates M. Thus, all we have to do is find the min-cost max-flow; the negative of its cost gives the unpopularity margin of M. To find it, we start with the max-flow representing M itself and augment negative cycles found using Goldberg's shortest-path algorithm [5]. □

Running time. To find each negative cycle, we run Goldberg's algorithm on our graph of $r + s$ vertices and m edges, taking $O(m\sqrt{r+s})$ time. Each cycle decreases the cost by at least 1 until we reach cost $-u$, so we find at most u cycles; then we perform one more failed search for a cycle. The bound follows. □

5 NP-hardness of finding least-unpopularity matchings

We now use a reduction from 3-satisfiability (3SAT) to prove that the problems of finding a least-unpopularity-factor matching and finding a least-unpopularity-margin matching for a given set of preferences are both NP-hard; it happens that the same reduction works for both problems. Abraham et al. [1] analyze preference sets with no ties separately from the general case of ties. We have had no reason to make this distinction so far, but the reduction will always generate preference lists with no ties in order to prove that even the no-ties versions of the problems are NP-hard.

The reduction converts an instance S of 3SAT to a polynomial-size preference set P and an *ideal* unpopularity factor. We will show that any tuple of truth values that satisfies S can be converted to a matching of P whose unpopularity factor does not exceed the ideal value, and vice versa; thus, the least unpopularity factor of P is the ideal value or less if and only if S is satisfiable. Clearly, if we had an algorithm to compute P's least unpopularity factor, we could use it to determine whether S is satisfiable. Therefore, computing the least unpopularity factor of a set of preferences is at least as hard as determining whether a 3SAT instance is satisfiable, so it is NP-hard. Computing an actual matching of that unpopularity factor is at least as hard still. This paragraph applies equally if "factor" is replaced by "margin"; in the description of the reduction, "unpopularity" refers to either "unpopularity factor" or "unpopularity margin".

5.1 Overview of the reduction design

The reduction, like most, builds P from *gadgets* that represent pieces of S. Each gadget will contain some *internal* people and positions and some *linking people*; there will also be *linking positions* that do not belong to any gadget. An internal person is willing to occupy only internal positions of her own gadget, but a linking person is also open to exactly one linking position, which is always her first choice. Any reassignment of the occupant of a linking position p is a demotion, which (from the perspective of voting) could just as well be to his last resort as into a gadget; thus, the dominance that can be achieved by replacing him depends only on the identity and state of the gadget providing the replacement, not on the states of any other gadgets. In other words, gadgets are isolated from one another unpopularity-wise; their only interactions are in *which gadget gets to fill each linking position*. Thus, we can analyze each gadget's contribution to the unpopularity of the matching separately as a function of which linking positions the gadget gets.

Motivated by this idea, we introduce several "models" of gadget, each of which is designed to enforce a certain constraint on which linking positions it must get by producing a low unpopularity if the constraint is satisfied or a higher one if it is not. To represent S, we start with a set of key linking positions representing its variables; the choice of which gadget gets each of these positions represents a tuple of truth values for S. We then add gadgets so that satisfaction of all of the gadget constraints is equivalent to satisfaction of S and let the ideal unpopularity of P be the low unpopularity that would result if every gadget's constraint were satisfied. Note that the unpopularity factor of the matching is the highest pressure produced by any gadget, while the unpopularity margins of separate gadgets generally add.

5.2 The gadgets

Our first type of gadget is the *box*. It consists of four internal positions, three internal people (i_1, i_2, and i_3), and three linking people (w, n_1, and n_2). Here is its structure, including the linking positions:

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

w is known as the *wide* person, and n_1 and n_2 are the *narrow* people. A box is satisfied, and produces a pressure of 2 and a margin of 1, if either the wide person or both narrow people get their linking positions; however, if both the wide person and at least one narrow person are denied their linking positions, a pressure of 3 and a margin of 2 result.

A *peg* consists of one internal position and three linking people, all of whom want the same linking position:

$$\begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{cc} x & l \end{array} \\ \begin{pmatrix} 2 & 1 \\ 2 & 1 \\ 2 & 1 \end{pmatrix} \end{array}$$

Its purpose is very simple: to always produce a pressure of 2 on l and provide a way to replace its occupant at margin 1.

A *pool* consists of two internal positions and three linking people with the following structure:

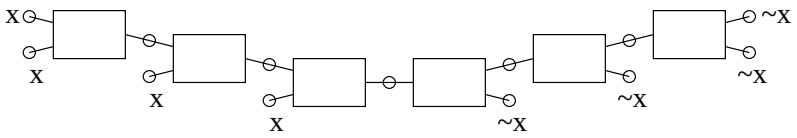
$$\begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{ccccc} x & y & l_{f1} & l_{f2} & l_{f3} \end{array} \\ \begin{pmatrix} 2 & 3 & 1 & - & - \\ 2 & 3 & - & 1 & - \\ 2 & 3 & - & - & 1 \end{pmatrix} \end{array}$$

If k of the people are denied their linking positions, the pool has one linking position with a pressure of k and can replace its occupant at a margin of $\max(k-1, 0)$. We want to use the pool to distinguish between two and three people being denied linking positions. To this end, we attach a peg to each linking position; then, all positions have pressure 2 and replacement margin 1, except when all three positions are taken by people from other gadgets, one of them develops a pressure of 3 and a replacement margin of 2.

5.3 Constructing the preference set

A box is a "two-for-one" device: if another gadget takes its wide linking position, it demands both narrow linking positions. For any k, we can construct a k-for-one device from $k-1$ boxes by identifying the wide position of each box after the first with a narrow position of the previous box. If we identify the ultimate wide positions of two such devices, we can get a u-for-v device for any desired u and v.

For each variable x_i of the 3SAT instance of S, we generate a many-for-many device whose two sets of narrow positions represent the references to x_i and the references to $\neg x_i$ in the clauses of S, respectively. The device for a variable x with four ordinary references and four negated ones could be diagrammed like this:



(Boxes represent boxes, circles represent linking positions, and lines represent linking people. Internal people and positions are not shown.) In a matching that obeys all the gadget constraints, we may assign all the linking people either to the right, filling the $\neg x$ positions and leaving the x positions open, or to the left, filling the x positions and leaving the $\neg x$ positions open. These two possibilities correspond to making x true or false, respectively. In either case, a linking position is left open if and only if the reference it represents evaluates to true.

Now, we add a pool for each clause of S and identify its three linking positions with linking positions of the variable devices according to the three references the clause contains. The pool's constraint is that at least one of its linking people receives a linking position, which means that at least one of the clause's references evaluates to true, i.e., the clause is satisfied. Thus, we can see that a matching that obeys all the gadget constraints corresponds to a tuple of truth values that satisfies every clause of S. We let the ideal unpopularity factor be 2 (regardless of S) and the ideal unpopularity margin be $6c - 2v$, where c is the number of clauses and v is the number of variables in S. This margin is the sum of the margin of 1 at satisfaction for each of the $3c - 2v$ boxes and $3c$ pegs.

5.4 Converting satisfying truth values to an ideal matching

The previous subsections have given an intuitive explanation of how the preference set P represents the original 3SAT instance S. In the next two subsections, we will show in more detail that a tuple of truth values satisfying S can be converted to an ideal matching and that no ideal matching exists if S is unsatisfiable, proving the correctness of the reduction.

Suppose we have a tuple of truth values $(t_1, \ldots, t_v)$ that satisfies S; we construct a matching M as follows. For each variable x_i, we decide how to assign the people in its device based on t_i. If t_i is true, we match each box on the non-negated (x_i) side according to the first table below, filling its wide linking position, and each box on the negated ($\neg x_i$) side according to the second, filling its two narrow linking positions. Each table's superscripts give the pressures generated by the box shown; a linking position may end up with a higher pressure from another gadget that is connected to it. In the first table, we let n_1 be a/the person whose linking position is shared with a pool (rather than a box); this is important so that the pressure of 2 exerted by the box on the position is no worse than that exerted by the attached peg. We then assign n_2 to his last resort.

	x^2	y^1	z^0	u^1	l_w^0	l_{n1}^2	l_{n2}^1
w	2	3	5	4	**(1)**	–	–
i_1	**(1)**	2	3	4	–	–	–
i_2	1	**(2)**	3	4	–	–	–
i_3	1	2	**(3)**	4	–	–	–
n_1	–	–	–	**(2)**	–	1	–
n_2	–	–	–	2	–	–	1

	x^2	y^1	z^0	u^0	l_w^1	l_{n1}^0	l_{n2}^0
w	2	3	5	**(4)**	1	–	–
i_1	**(1)**	2	3	4	–	–	–
i_2	1	**(2)**	3	4	–	–	–
i_3	1	2	**(3)**	4	–	–	–
n_1	–	–	–	2	–	**(1)**	–
n_2	–	–	–	2	–	–	**(1)**

Observe that each linking position connected to the device is filled exactly once, except for the positions representing references to x_i; furthermore, no pressure greater than 2 is generated. If t_i is instead false, we use the same construction but with the two sides of the device switched. Either way, exactly those linking positions that represent references evaluating to true are left open.

After we match all the variable devices, we assign each pool linking person to his/her linking position if it is available, or otherwise to the best available position in his/her pool. Since the t_i satisfy S, at least one linking person from each pool will get a linking position, so each pool only needs to accommodate at most two people. This table shows how a pool can hold two people (the pressure superscripts consider the attached pegs as well as the pool itself):

$$\begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \end{array} \begin{array}{c} \begin{array}{ccccc} x^1 & y^0 & l^2_{f1} & l^2_{f2} & l^2_{f3} \end{array} \\ \begin{pmatrix} (\mathbf{2}) & 3 & 1 & - & - \\ 2 & (\mathbf{3}) & - & 1 & - \\ 2 & 3 & - & - & (\mathbf{1}) \end{pmatrix} \end{array}$$

It should be clear that moving more people to linking positions does not increase the pressures.

The argument that M has the ideal popularity factor 2 is nice and easy: nowhere in the above did we incur a pressure greater than 2. Showing that M has the ideal unpopularity margin $6c - 2v$ takes more work. We will appeal to the following lemma:

Theorem 5.1. *If M is a Pareto efficient matching, the number of filled positions in M that have pressure 2 or greater is an upper bound on the unpopularity margin of M.*

Proof. Let N be a matching that dominates M by the greatest margin, which is M's unpopularity margin. As in the proof of Theorem 3.4, we modify N so that all demotions are to last resorts and observe that $M \oplus N$ becomes a collection of vertex-disjoint paths, each of which promotes one or more people and then demotes one person to her last resort. (No path can consist of just a demotion because it could be dropped, making N dominate M by a greater margin.) Consider a path W that promotes k people; it contributes $k - 1$ to N's dominance margin over M. For each i, let p_i be the position filled by the ith promotion. If, for some i, we replace the portion of W after p_i with a demotion of the occupant of p_i to his last resort, we get a path demonstrating that the pressure of p_i is at least i. In particular, $p_2, \ldots, p_k$ have pressure at least 2. By charging W's dominance-margin contribution of $k - 1$ to these $k - 1$ positions, we see that the total dominance margin is at most the total number of filled positions of pressure 2 or greater, as desired. □

Now we just count the positions of pressure 2 in M. Every linking position connected to a pool has pressure 2 by virtue of its peg; there are 3 such positions for each clause for a total of $3c$. Each box's internal position x also has pressure 2. S contains $3c$ variable references, and each gets its own box except that the two endmost references on each of the 2 ends of each of the v variable devices share a box; thus there are $3c - 2v$ boxes. This makes a total of $6c - 2v$ positions of pressure 2, and one can check that there are no others. By Theorem 5.1, M has unpopularity margin at most $6c - 2v$, as desired.

Although it is not needed for the correctness of the reduction, we mention that one can recover a satisfying tuple of truth values from an ideal matching M by looking at who occupies the central linking position of each variable x_i's device. We set t_i to true or false as that position is filled by the ultimate wide person from the non-negated or negated side of the device, respectively.

5.5 Nonexistence of ideal matchings for unsatisfiable 3SATs

For the other direction of the correctness proof, suppose that S is unsatisfiable and let M be an arbitrary Pareto efficient matching of P; we will show that M has neither the ideal unpopularity factor nor the ideal unpopularity margin. It should be clear from previous discussion in this section that, since S is unsatisfiable, M cannot satisfy all the gadget constraints, but we still must show that this results in non-ideal unpopularities.

M must dissatisfy either a pool or a box. Suppose first that it dissatisfies a box, i.e., both the wide person w and at least one narrow person (say n_1) are denied linking positions. The four people w, i_1, i_2, and i_3 are all open exactly to the four positions x, y, z, and u. If one of the positions were empty, one of the people would be at her last resort and eager to fill it, a Pareto improvement; thus M must fill all four positions. If n_1 is at his last resort, then we could promote him to u, its occupant to y, and its occupant to x, demoting the occupant of x to his last resort. Otherwise, one of w, i_1, i_2, i_3 is at her last resort, and we could promote her to z, its occupant to y, and its occupant to x, again demoting the occupant of x. Either plan gives a dominance margin of 2 and shows that the pressure of x is at least 3.

On the other hand, if M dissatisfies a pool (by denying all three of its people their linking positions), then one of the people must be in x, one must be in y, and the third (call him p) must be at his last resort. We could promote p to y, its occupant to x, and its occupant to his linking position, demoting its previous occupant; this plan also gives a pressure of 3 (this time on the linking position) and a dominance margin of 2.

The pressure of 3 is enough to show that M fails to achieve the ideal unpopularity factor of 2. Unforunately, we lack a nice analogue to pressure for margins, so to show that M's unpopularity margin is non-ideal, we will construct a matching N that dominates M by a margin of more than $6c - 2v$.

Consider a peg in M; it may or may not get its linking position, but either way, x must be filled for Pareto efficiency, and at least one person is left at her last resort. Starting from M, we "cycle" each peg by promoting a last-resort person to x, promoting the occupant of x to the linking position, and demoting the occupant of the linking position. There are $3c$ pegs, each with two promotions and one demotion, so N dominates M by a margin of $3c$ so far.

Now we turn to the boxes. In each box, the three people i_1, i_2, and i_3 are all eager to have one of the three positions x, y, z, so M must fill those positions for Pareto efficiency. In N, we promote whoever occupies z (it could be w rather than an i_j) to y and its occupant to x, demoting the occupant of x. There are $3c - 2v$ boxes, so this gives us an additional dominance margin of $3c - 2v$ for a total of $6c - 2v$, the ideal margin.

To reach a non-ideal margin, we make use of the dissatisfied gadget. If a pool is dissatisfied, let p be the position on which it exerts pressure 3. Instead of cycling the peg attached to p (which contributes 1 to the dominance margin), replace the occupant of p according the pool's plan (above), which contributes 2 to the dominance margin. Similarly, if a box is dissatisfied, use its margin-2 plan instead of the margin-1 plan of the previous paragraph. Either way, the total dominance margin of N over M increases to $6c - 2v + 1$, so M cannot achieve the ideal unpopularity margin and the proof is complete.

5.6 Conclusion

By means of the reduction, we have proved the following theorem:

Theorem 5.2. *The least unpopularity factor and least unpopularity margin are both NP-hard to calculate for arbitrary preference sets. Consequently, it must be NP-hard to find a matching of minimum unpopularity factor or margin for a given preference set.*

Web page

I have put up a Web page at `http://mattmccutchen.net/lufm/` with some additional materials related to this work. Notably, it has the `popular-matcher`, a collection of Java code containing

the algorithms to find the unpopularity factor and margin, the conversion from 3SAT to preference sets, a number of heuristic algorithms for finding low-unpopularity matchings that I tested before discovering that the problem was NP-hard, and various other items I found useful during my work. Others are welcome to use this software to experiment further with the ideas from this paper.

Acknowledgments

I would like to thank Samir Khuller, my advisor, for introducing me to previous work on matching with one-sided preferences; Brian Dean for suggesting the use of Goldberg's algorithm in the algorithm to calculate the unpopularity factor; and Dr. Khuller, Bobby Bhattacharjee, Glenda Torrence, Nancy Zheng, and others for suggesting improvements to this paper and its precursors.

References

[1] D. Abraham, R. Irving, T. Kavitha, and K. Mehlhorn. *Popular matchings.* Proceedings of SODA 2005: the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, 424–432.

[2] D. Abraham and T. Kavitha. *Dynamic matching markets and voting paths.* Proceedings of SWAT 2006: the 10th Scandinavian Workshop on Algorithm Theory, 65–76.

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

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

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

[6] R. Irving, T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch. *Rank-maximal matchings.* Proceedings of SODA 2004: the 15th Annual ACM-SIAM Symposium on Discrete Algorithms, 68–75.

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

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

[9] A. 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.