Making Good Matches

Problem 1 Victoria, Wyatt, Xena, Yetta, and Zeus are applying for jobs A, B, C, D, and E. Here are their preferences, ranked from most preferred job (or employee) to least:

person	preferences		employer	preferences
v:	B A D E C		A:	z v w y x
w:	D B A C E		B:	x w y v z
x:	B E C D A		C:	w x y z v
y:	A D C B E		D:	v z y x w
z:	B D A E C		E:	y w z x v

How many different ways can you match the people to the jobs in pairs? How can we match the the people to jobs in such a way as to optimize . . . well, what do we want to optimize?

Problem 2 With n people, and n employers, how many different tables of preferences could there be? [For simplicity, we are not worrying about permutations of the people.]

1 Once we know what "stable" means

Problem 3 for the table of preferences listed above, can you find a stable matching? How well, on average, do the job-seekers do? (that is, do they, on average, get their favorite job? their third favorite job? or is the average some non-integer?) How well, on average, do the employers do?

Problem 4 A student at a recent math circle session suggested this approach: Each job-seeker gives a numerical ranking of the employers from 1 (best) to n (worst) and the employers similarly rank the job-seekers. Each possible pair is then scored with the sum of the two rankings and we find a matching with the minimum possible sum of all such ranks. Is the resulting pairing guaranteed to be stable?

Problem 5 Given a fixed matching of n people and n employers, what is the largest number of blocking pairs which could happen? Can you come up with a table of preferences and a matching that results in that many blocking pairs?

Problem 6 Is it possible to find a set of people and employers with preference orderings that has exactly 1 stable matching? exactly 2 stable matchings? Pick a small number n (say $n = 3$, $n = 4$, or $n = 5$). With n people and n employers, can you find a set of preferences for them that results in n different stable matchings? $2n$? n^2? 2^n?

Problem 7 Find all stable matchings for this preference ordering

person	preferences		employer	preferences
v:	E D C B A		A:	v w x y z
w:	A E D C B		B:	w x y z v
x:	B A E D C		C:	x y z v w
y:	C B A E D		D:	y z v w x
z:	D C B A E		E:	z v w x y

Problem 8 (Once we have settled on an algorithm) Given the algorithm we are using, does it matter the order in which the procedure is run? That is, if we allow some of the job-seekers to be matched up (provisionally) first and only then the remaining job seekers begin the process, is the result any different than if they all began at the same time?

Problem 9 If we assume 100 job seekers, and 100 jobs (and assume a random assignment of preferences), about how well will the job seakers do? About how well will the employers do?

Problem 10 How should we handle things if there is one more person than job? (this example is randomly generated)

person	preferences		employer	preferences
u:	D B E C A			
v:	A C E D B		A:	x v u y z w
w:	B C D E A		B:	x w u z v y
x:	D C B E A		C:	z u y w x v
y:	C E A B D		D:	y x z v w u
z:	B D A E C		E:	z w y u v x

How well do the job-seekers do on average? How well do the employers do? (Which algorithm are we using). Could the person who ends up without a job have done better had he or she changed his preference ordering before the assignments were made?

Problem 11 If there were 10 job-seekers, and 10 employers, and each job-seeker only listed their five most preferred choices, what could go wrong?

Problem 12 Would it ever make sense for a job-seeker to lie (that is, to list a preference ordering other than his or her true preference ordering)? Would it ever make sense for an employer to lie?