School Matching Algorithm Implementation

My code implements a somewhat simplified version of the algorithm Alvin Roth designed for assigning students to high schools in the New York City public school system. Roth's algorithm is a variant of the "Deferred Acceptance Algorithm" designed and analyzed by Gale and Shipley. This comes from a class of problems in mathematics and economics known as "Stable Matching Problems". The easiest example, and of which Gale-Shipley is an extension is the so – called "stable marriage problem". Suppose there are an equal number of men and women, and they are looking to form couples. A set of couples is *stable* if there is no man and no woman who prefer each other to their current partners. Given a list of preferences over the women submitted by each man, and a list of preferences over the men submitted by each of the women, an algorithm exists that can find a stable matching. In the algorithm, men propose, and women evaluate proposals. An important result, whose proof I will omit since it is easily found, is that the matching is optimal for those that propose. That is, each man gets the most preferred stable female partner. The Deferred Acceptance Algorithm below is an extension to case in which instead of marriages, we consider applicants applying to colleges, which have many spaces open.

Description of Algorithm as applied to schools:

1) In New York City, students applying to high schools create a list of preferences with a least 5 schools, and at most 12. In my test implementations, the numbers are less than that, since only use 8 students and 5 schools. Preferences must be **strict**.

2) Schools receive application materials for all students who have put the school on their list. Schools **do not see** students' preferences. Schools may rate some students as unacceptable. In the original Gale-Shipley formulation, students who don't meet standards are simply assumed not to apply. All applicants who have met standards are ranked by the school. Preferences must be **strict** .Ranking are submitted to the system.

3) The algorithm starts. Students are kept in queue.

Step 1: Examine first choice school for student.

i) If student not acceptable to school, return to queue.

ii) Else If acceptable and seats available, tentatively place.

iii) Else If acceptable and quota filled, compare to matched students

1) If preferred, match. Return least desirable student to queue.

2)If not preferred, return proposing to queue.

Step 2: Repeat step 1 for 2nd choice of students who were no matched to 1st choice

Step N: Loop step 1. Algorithm terminates when queue is empty (no more rejections given). Schools finally matched to students on list after last step.

The main difference between my implementation and the case of NYC schools is that I only have one type of school, in the sense that each school uses some set of criteria to manually rank applicants and submit their rankings to the system. In NYC, there are some schools that have other methods of selection. Some schools must give more weight to students from a certain geographic zone. Other schools are so-called EdOpt Schools in which 50% of places are based on a school's preference, and the other 50% distributed by lottery with a 16/68/16 distribution of reading exam scores. In NYC, the algorithm handles this case by assigning randomized preferences to students. Including such schools does not really add extra insight, but adds significant difficulty, so this is not part of my algorithm. If such an extension were to be desired however, this could be accomplished by writing a derived class of School with additional and/or modified methods.

An important issue in terms of performance is my queue of students. Given a data set of realistic size, with 100,000 students or so, and several hundred schools, checking one student at a time is a serious bottleneck. Ideally, the program would be parallelized so that many students would be removed from the queue and checked simultaneously. Issues such synchronized writing to the queue presented enough technical challenges that I did not implement this.

As a final note, I simply accept preferences as they are submitted. Game theoretically, for those proposing, stating true preferences is a dominant strategy. Although in the marriage case, stating true preferences is also a dominant strategy for the women, this is not necessarily true in the Deferred Acceptance version. However, the unpredictability to other's choices and other such constraints make it exceedingly difficult to "game" the system in practice. Regardless, my code only works with preferences as they are submitted, and how they are determined is irrelevant to my implementation.

References:

http://economics.mit.edu/files/3024 (Roth NYC Schools paper).

http://www.cs.princeton.edu/courses/archive/spr05/cos423/lectures/01stable-matching.pdf

http://web.stanford.edu/~alroth/papers/92_HGT_Two-SidedMatching.pdf