Incremental Stable Roommates Algorithm For Clustered Input Attributes

BHARGAVI KARTHIK, CARMINA FRANCIA, HALEIGH WALKER & NIKITA TRIVEDI,

University of Texas at Austin

The Stable Roommate Problem in the domain of computer science involves pairing elements of a set as roommates, based on their individual preferences. One common and useful variation to this problem includes finding a new matching given an existing matching and a few students joining or leaving the group. This paper proposes a new algorithm to solve this incremental problem with a focus on optimizing for a commonly occurring situation where the preference list of the students have a clustered attribute. The proposed algorithm efficiently partitions the input data into two sets of students, a pool of happy students of existing matching who are not affected (no blocking pairs) by the incremental change and a pool of remaining students who make up the unhappy pool. The algorithm then proceeds to augment the existing matching by matching students in the unhappy pool using the multiple phases of Irving's algorithm either resulting in a final stable match or concluding that a stable match is not possible for the given input. We implement this algorithm using Java and evaluate it with multiple test cases consisting of clustered attributes. The algorithm achieves 27 percent improvement in execution time for a group of 100 students with five clusters and up to 92X speedup for best case scenarios using 1000 students consisting two skewed clusters.

1 Introduction

The Stable Roommate Problem [2] is a variation of the Gale-Shapely Stable Marriage Problem. Irving's algorithm [1] is an efficient method for finding solutions to the Stable Roommate problem when they exist for a given set. However, one commonly observed scenario is that once a stable matching is achieved for a given set, there could

2018. XXXX-XXXX/2018/11-ART1 $15.00

https://doi.org/10.1145/nnnnnnn.nnnnnnn

be increments to the set in terms of new students joining and existing ones leaving the set. This incremental change can potentially break all the existing matches or be less invasive and achieve a stable match re-using a significant portion of the existing matches. In many cases, it is not uncommon for the input set to have clustered preference attributes that will result in the possibility of reusing some of the existing matches. For example, a group of students can have clustered preferences based on their ethnicity, gender, field of study etc. Consider the case where a new student joins one such cluster thereby requiring re-matching only for the students in this cluster. Leveraging this practical skew, we propose a new algorithm that specializes in significantly speeding up the execution time for incremental settings in this category of use cases. To maintain an even sized set, we assume that elements (say m) will be added or removed in multiples of two and m « n, where n is the total number of students. While Irving's algorithm has a time complexity of O(n^2), our modified algorithm incorporating the incremental setting for clustered attributes has a time complexity of O(nk), where k is the cluster size. When the input problem size scales up and we have k « n, the performance improvement from using our algorithm over the naive Irving is huge. The main contributions of this paper include: The main contributions of this paper include:

(1) Proposal of the incremental stable roommate algorithm optimized for clustered attributes

(2) Implementation of the newly proposed algorithm

(3) Implementation of Irving's algorithm to use as baseline

(4) Functional validation using hand-crafted test cases

(5) Performance evaluation using large test cases (100x100, 1000x1000) against naive Irving's algorithm

The paper is structured as follows: in Section 2 we describe the algorithm we have implemented in detail. In Section 3 we discuss design alternatives. In Section 4 we review our performance result. In Section 5 we draw conclusions for the paper.

2 Project Description

2.1 Introduction to the Problem

The Stable roommates problem (SRP) involves having a set of '2n' number of participants, each ranking the other 2n-1 participants in their order of preference. A matching M consists of finding n pairs of participants, which is stable, only if there are no two participants x and y, each of whom prefers the other to their current

partners in M. Such a pair is said to be a blocking pair with respect to M and would lead to no stable matching in the given set. Hence, there might be no stable matching that exists for a certain set of participants and their preferences.

To find a stable matching, first take a look at the basics of the Irving's algorithm. The Irving's algorithm mainly consists of two phases.

2.2 Phase I

2.2.1 Proposal

Let us start with 2n number of participants in a set. The first phase consists of each participant ordering all the other participants by preference (known as their preference lists). Participants then propose to each person in their list, continuing to the next person if their proposal is rejected in the following scenarios:

(1) A participant could reject a proposal if they already have a better proposal from someone they prefer more or

(2) A participant could also reject a proposal that was previously accepted by them, if they are proposed by someone higher on their preference list later during this phase.

If a participant is rejected by all the people in the list, it indicates that no stable matching is possible, otherwise, Phase I would result in each participant holding a proposal from one of the others.

2.2.2 Reducing The List

If every person holds a proposal, the next step is to reduce the preference list to rule out the worst matches for every participant. Consider two participants p and q where q holds a proposal from p, then,

We remove all those participants (say x) from q's list that are lower in the order of preference than p.

For each participant x that is removed from q's list, we also remove q from x's list to make sure that q is first in p's list and p is last in q's list. Hence, we make sure that the rejections are symmetrical.

The table of participants that we get after we have reduced our preference list is called a stable table. If the reduced list after Phase I table gives us exactly one individual for every participant, then this gives us a matching. Otherwise, we enter the Phase II of the algorithm to find a rotation in the stable table.

2.3 Phase II

To find a rotation in a stable table, start with a person pi in the table that has at least two individuals in their reduced list:

Define q_i to be the second preference of p_i. Define p_i +1 as the last preference of q_i. Repeat the above steps until the sequence repeats some p_j at which point a rotation or a cycle is found. Reject from the table, all the pairs of ($q_i \rightarrow$ p$_i$ +1)

If at any point, there is a participant with no person left in the reduced list, there exists no stable matching. However, if there exists any person with more than one person in their reduced list, repeat this process until everyone has only one person left. Then, you have a stable matching of n pairs of participants.

2.4 Incremental Stable Roommates Algorithm For Clustered Input Attributes

Our algorithm focuses on providing a solution for efficiently finding stable matches (if exists) for an incremental setting with a focus on optimizing for scenarios where the preference list of the students have a clustered attribute. The proposed algorithm efficiently partitions the input data into two sets of students, a pool of happy students of existing matching who are not affected (no blocking pairs) by the incremental change and a pool of remaining students who make up the unhappy pool. The algorithm then proceeds to augment the existing matching by matching students in the unhappy pool using the multiple phases of Irving's algorithm either resulting in a final stable match or concluding that a stable match is not possible for the given input.

2.4.1 Run-time Complexity

Our proposed algorithm incorporating the incremental setting for clustered attributes has an overall time complexity of O(nk), where k is the size of the various clusters within the whole set and n is the total number of students. This includes:

1) In the section covering lines 4 to 18 in Algorithm 1, one can note that we do less than n comparisons (cardinality of students in matching) for k+m (m is the number of students joining/leaving) elements that get added to the unhappy pool resulting in a time complexity of O($n(k + m)$) which can be approximated to a time complexity of O(nk).

2) For line 19, to construct the new preference list, we remove n-k rows for n-k happy students from the preference

Algorithm 1 Pseudocode for Incremental Setting

```
1: Base Case: Run Irving's algorithm with the original n number of students
2: Assuming that a stable matching is achieved, keep the (n/2) matchings saved
3: Incremental Algorithm:
4: if even number of new students join then
5:     for each student s that is joining do
6:         AddToUnhappyPool(s)
7: if even number of students leave then
8:     if not all leaving students are matched within themselves then
9:         for every student s who lost their partners do
10:            AddToUnhappyPool(s)
11: function ()AddToUnhappyPool(Students)
12:     FOR EACH PAIR (s1, s2) IN THE MATCHING DO
13:         IF BOTH ARE HAPPY WITH THEIR CURRENT ROOMMATE COMPARED TO S THEN
14:             KEEP THE MATCHING (s1, s2) INTACT
15:         ELSE IF AT LEAST ONE OF THEM IS NOT HAPPY WITH THEIR CURRENT ROOMMATE COMPARED TO S THEN
16:             AddToUnhappyPool(s1)
17:             AddToUnhappyPool(s2)
18:             REMOVE THE PAIR (s1, s2) FROM MATCHING
19: CONSTRUCT A NEW PREFERENCE LIST BY EXCLUDING THE MEMBERS IN MATCHING
20: RUN BASIC IRVING'S ALGORITHM USING THE NEW PREFERENCE LIST
21: IF IF STABLE MARRIAGE ACHIEVED THEN
22:     AUGMENT THE NEW MATCHINGS TO THE ORIGINAL LIST TO PROVIDE THE FINAL STABLE MATCHINGS M
23:
```

list and then remove these n-k students from the preference lists of the remaining k students in the preference list. This incurs a time complexity of (n-k) + (n-k) * k which can be approximated to a time complexity of O(nk). 3) For line 20, the time complexity is O(k^2) to run the basic Irving's algorithm with the remaining k students. Given that k < n, this can also be approximated to O(nk).

If implemented using data structures like hashmaps and arraylists, most of these operations can be done quite efficiently compared to our baseline of running the Irving's algorithm from scratch for each of the incremental input.

3 Design Alternatives

The Stable roommate algorithm focused on having an even number of participants. Suppose only a single person enters or leaves the set. This results to an uneven number of participants which is a special case for a Stable Roommate Problem. In order to handle this, below is an alternate solution:

For instance, one person enters or leaves the current set after an initial stable matching. The idea is to let the new participant make proposals according to his/her preference list. The participant will go through a proposal-rejection process. The current set will either accept or reject. If nobody from the current set accepts the proposal, the matching stays the same because everybody in the current set thinks they have a better partner, hence no blocking pair. The new participant is left alone. On the other hand, say we have (P1, P2) matching. P1 decides to accept the new participant. P1 and the new participant becomes a pair. P2 is now by itself. To ensure there's no blocking pair, P2 will have to go through the same proposal-rejection process. Because it is possible to get caught in a cycle resulting in an infinite loop, we either need a mechanism to track repeating rejections resulting in a cycle or use phase three of Irving's algorithm to eliminate cycles. By doing this, we guarantee that the unmatched participant does not cause a blocking pair for the remaining matches. The problem with an uneven number of participants is that it is not possible to have a perfect stable matching; however, there exists a feasible solution that maximizes the number of stable matches. This design alternative can save some computations for an incremental stable roommate algorithm over naively invoking Irving's algorithm for each incremental input. But, we believe that coming up with an algorithm to solve the incremental problem for a commonly occurring input attribute like clusters can be more generally useful, showcase bigger speedups and also serve as an interesting aspect to explore as part of this paper.

4 Evaluation and Results

In order to evaluate to the soundness of our algorithm in terms of functionality, we came up with several test cases covering the various negative and positive scenarios. The test cases were handcrafted in such a way that

stable matching can be achieved, stable matching is not possible, leads to instability during the various phases of Irving's basic algorithm. Also for testing the different modules in the new algorithm for the incremental setting, we constructed test cases with varying proportions of number of happy couples with respect to the total number of original matchings.

While evaluating the performance of our algorithm, we noticed a lot of run-to-run variations when using smaller test cases (6 X 6). This was probably due to the variability introduced by the JIT compiler in Java, owing to the short run times. Hence, to avoid these variations we had to generate larger test cases (like 100 X 100, 1000 X 1000). However, compiling such large test cases that would also yield stable matching was a complex task which cannot be done manually. We created a Java program that would randomly generate large test cases and also validate if these test cases would yield a stable matching, thus enabling us to effectively do the performance evaluation. These test cases were designed in such a way to measure the performance of the algorithm with various cluster sizes.

A test case with 100X100 matrix (as bundled with code) was generated with five clusters each with 20 students and their preference list randomly distributed. Two incremental settings were applied to this case as following,

1) Two new students joined and were introduced randomly to one of the five clusters

2) The same two students leave The algorithm achieved 27 percent improvement in execution time as shown

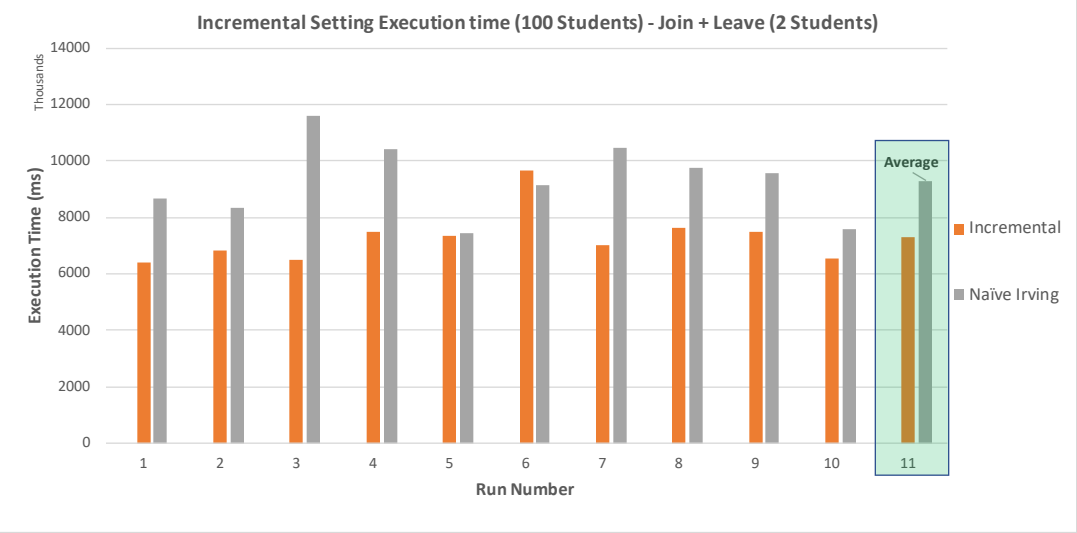

Fig. 1. Incremental setting execution time for 100 students and 5 clusters

in Figure 1 over invoking each incremental setting with Irving's basic algorithm repeatedly.

Another test case with a 1000 X 1000 matrix was generated consisting of one large cluster with randomized preference list and applying an incremental change of two new students joining who are not preferred much by the existing students resulting in two skewed clusters. This is close to a best-case scenario which can show case an upper bound on the potential speedup that can be achieved by our algorithm over Irving's algorithm. As shown in Figure 2 the algorithm achieved an average speedup of 92X.

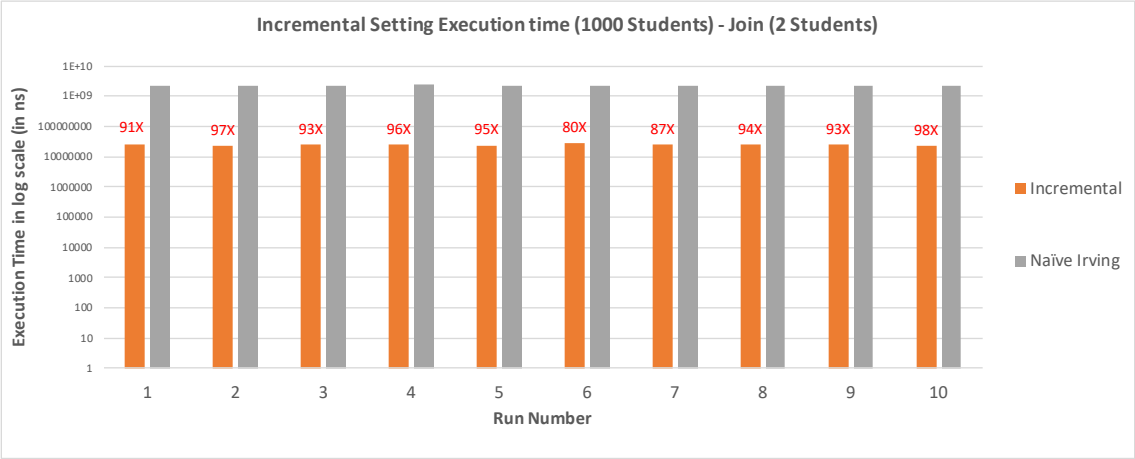

Fig. 2. Incremental setting execution time for 1000 students (best case scenario)

5 Conclusion

In this paper, we are proposing an efficient algorithm that can provide a solution for the existing Stable Roommates Problem under incremental conditions such as students leaving or joining the set. The main focus area of the algorithm is to optimize for a frequently observed scenario of students preferring to stay within a tightly coupled groups of varied sizes, thus exhibiting a clustered attribute in the provided preference list. This is achieved by isolating the matches that are not affected by the increment and finding matches for only those remaining, whose cardinality will be significantly low compared to those who are not affected. Our Java implementation of this approach was evaluated using test cases containing clustered attributes and the results show a 27 percent improvement in execution time (100 elements, five clusters) and up to 92X speedup for best case scenarios (1000 elements, two skewed clusters).

References

[1] Robert W Irving. "An efficient algorithm for the 'stable roommates'problem". In: *Journal of Algorithms* 6.4 (1985), pp. 577–595. ISSN: 0196-6774. DOI: https://doi.org/10.1016/0196-6774(85)90033-1. URL: http://www.sciencedirect.com/science/article/pii/0196677485900331.

[2] Nathan Schulz. *The Roommates Problem Discussed Roommates Problem*. 2008. URL: http://www.math.uchicago.edu/~may/VIGRE/VIGRE2008/REUPapers/Schulz.pdf.