Enhancement of Irving's Algorithm with Autocomplete Feature

Karen Claire Morfe[1], Ulysis Agustin[1], Mark Christopher Blanco[1], Dan Michael Cortez[1], Antolin Alipio[1]

[1] kcgmorfe2018@plm.edu.ph
[1]Computer Science Department, College of Engineering and Technology
Pamantasan ng Lungsod ng Maynila (University of the City of Manila)
Intramuros, Manila 1002, Philippines

Abstract

Irving's algorithm is the solution Robert W. Irving proposed to Gale and Shapley's Stable Roommate problem. It determines if there is a possible stable match in each set using a list of preferences, and if it is possible, finds the stable matches. However, incomplete preference lists would increase the chances of having one participant become unmatched. This paper introduces an enhanced version of this algorithm that uses the addition of descriptive lists to fill in the vacancies in the incomplete preference list. The conclusion is that it is more efficient to fill the vacancies according to approximate preference rather than leaving the lists as incomplete to create more stable matches.

Keywords: Irving's Algorithm; Matching; Stable Roommates

1. Introduction

Irving's algorithm is a matching algorithm that can help solve stable matching problems. This algorithm determines if there is a match in each set using a list of preferences, and if it is possible, finds the stable match. Irving's algorithm shows that in general cases that are being examined, properties of bipartite stable matchings will remain. In addition, there are existing graphs with preferences that do not include stable matchings and the researcher hopes that one algorithm that finds a stable matching will exist as well as stating that there is no stable matching, Szestoplaw, 2010. With this, in 1985, Robert Owen provided Irving's algorithm. This algorithm consists of two phases.

The first phase wherein there will be a proposal. In addition to this, the first phase also has the possibility that a person can be rejected by everyone else. If this happened, the person who matched themselves would rank last in their preference list making it a not stable match. For phase two, there are potentially multiple partners left for a given partner. With this, phase two will find a useful cycle among those who are matched from phase 1 and then remove all the possibility until a stable matching is achieved or declaring that there is no stable matching, Yue et. al, 2016.

Irving's algorithm is extremely useful because unlike its similar algorithm, the Gale-Shapely algorithm, it can offer and receive proposals, Szestoplaw, 2010. A good example where Irving's algorithm is used was the stable roommate's problem. Its goal is to find a stable matching wherein each agent has preference over his/her roommate, Boehmer et. al, 2020. Although Irving's algorithm caters stable and unstable matching, there is a possibility wherein the list of each agent is not complete. With this, an enhancement of Irving's

algorithm is introduced.

2. Existing Irving's Algorithm

2.1. *The problem in Irving's algorithm*

One of the problems in Irving's algorithm is that it needs to have an ordered list of n-1 for each participant's preference list. In Cseh, et al., 2017, mentions that "agents may find it difficult to rank a large number of alternatives in strict order of preference. One natural assumption, therefore, is that preference lists are short." If applied to real world scenarios such as assigning students that will share dorms, most students would not know the others that are participating or may not know them well enough to make a proper preference. That would mean that either the student will create an incomplete list consisting of only those student knows or randomly ordering those that student does not know well. In the case that each student must have a partner, either of these scenarios can create dissatisfaction with their assigned roommate despite following the given ordered list. If the students submit an incomplete list, they have a higher chance of remaining unmatched and randomly ordering could assign the student to someone that they could not actually get along with. Chen, 2019, also states that varying from application to application, the preferences of the agents in Stable Roommates could be incomplete which would then mean not every agent could be an acceptable partner to every agent. This would make it harder to find stable matches.

2.2. Pseudocode of Irving's algorithm

```
while (there are unmatched agents):
    i = smallest value such that pi is unmatched
    p = unmatched agent
    q = agent in preference list of p that has not rejected p previously
    pi proposes to q:
        if q has never received a proposal:
            q accepts pi
        else:
            if q prefers proposal of pi compared to current proposal:
                q accepts pi
            else:
                q rejects pi
    for all accepted proposals (p,q):
        x = agent in preference list of q
        q rejects all x that is lower on the preference list compared to p

    if there is more than 1 agent left in any preference list of p:
        for all cycles (c1,..,cn) and associated second preferences (d1,..,dn)
            di = second preference of ci
            ci+1 = last preference of di
            cn ∈ [ci,...,cn]:
                for i in range (1, n-1):
                    di rejects ci+1
```

no stable match exists if an agent had been rejected by all the other agents and the algorithm will end.

3. Enhanced Irving's Algorithm

3.1. Enhancement of the Algorithm

In order to achieve the goal of providing lesser chances of unstable matches occurring and also to provide an easier way of ranking participants despite not knowing them, this paper proposes adding descriptive lists that are associated with the participants involved. The information in the additional lists will serve as the basis for the algorithm to autocomplete each incomplete list. To fill in the vacancies, the qualities the participant is looking for will be compared to the qualities that the other participants have and then filled in order of whose is the highest match to the least. As stated by Chen, 2019, "even if a given Stable Roommates instance admits a stable matching, this solution may not be unique, and there might be solutions with which the agents are more satisfied than with others and thus, are more desirable than others. Given these two facts, it makes sense to consider two types of optimization variants for Stable Roommates: In one type one would want to compute stable matching that optimize a certain social criteria; in the other, one would want to compute matchings with optimal distance or closeness to stability". The addition of descriptive lists falls under the stable matchings that optimize a certain criteria. Moreover, these descriptive lists would not count as ties and therefore the algorithm remains NP and not NP-hard. In Roth, et. al, 2012, "Stable Matching: Theory, evidence, practical design" , they mentioned that having an unrestricted trading is a key presumption underlying the concept of stability. In connection to the enhanced algorithm, the statement given by Roth, et. al, 2018, proves that stable matches can also be achieved if incomplete lists are being accepted unlike in the original Irving's algorithm wherein incomplete lists are not already accepted in its first phase. Another paper written by Chen, et. al, 2018, entitled "A Theory of Stability in Matching with Incomplete Information", stated that although that the prevailing assumption is to have a complete information, incomplete information is seen in matching markets. In line with our study, even though incomplete list can produce an unstable matching, it is the reality that not all lists are complete. As a result, incomplete list will be accepted in the enhanced algorithm that the original algorithm didn't do. The enhanced algorithm will autocomplete the incomplete list to have a stable matching.

3.2. Pseudocode of Enhanced Algorithm

```
for all preference lists of p:
    i = smallest value such that pi is unmatched
    a = list of qualities that the agent is looking for
    b = list of qualities the agent has
    p = unmatched agent
    o = agent that is not p
    if length of preference list of pi is less than n:
        f = list of oi's weighted averages
        for all qualities in b:
            k = quality in b
            s = arbitrary integer
            if ki exists in a:
                if index of ki is equal to index of b1:
                    weight of ki = s / (s* length of b) * 0.45
```

```
                                    if index of ki is equal to index of b2:
                                        weight of ki = s / (s* length of b) * 0.35
                                    if index of ki is equal to index of b3:
                                        weight of ki = s / (s* length of b) * 0.2
                                    t = t + weight of ki
                weighted average of q = t
                store weighted average of all q in f
                sort f from greatest to least according to weighted average
                for all y listed in f:
                        y = agent listed in f
                        if yi is not in the preference list of p:
                                append yi to preference list of p
    while (there are unmatched agents):
        i = smallest value such that pi is unmatched
        p = unmatched agent
        q = agent in preference list of p that has not rejected p previously
        pi proposes to q:
                if q has never received a proposal:
                        q accepts pi
                else:
                        if q prefers proposal of pi compared to current proposal:
                                q accepts pi
                        else:
                                q rejects pi
        for all accepted proposals (p,q):
                x = agent in preference list of q
                q rejects all x that is lower on the preference list compared to p

        if there is more than 1 agent left in any preference list of p:
                for all cycles (c1,..,cn) and associated second preferences (d1,..,dn)
                        di = second preference of ci
                        ci+1 = last preference of di
                        cn ∈ [ci,...,cn]:
                                for i in range (1, n-1):
                                        di rejects ci+1
    no stable match exists if an agent had been rejected by all the other agents and the algorithm will end.
```

4. Methodology

In this study, experimental research design is used. The researchers used this type of research design in order to control the factors that might affect the result of the study. With this, the researchers try to determine what will occur on the following processes.

Experimental research design is used to establish a relationship between the situation's cause and effect. In addition to that, this research design also observes the impact caused by the independent variable on the dependent variable. Independent variables are being manipulated in order to know the change in results for

the dependent variables. With the given explanation regarding the experimental research design, it is a highly practical method of research since it can contribute to solving a problem.

With the help of other resources such as articles, journals, etc., the methodology created by the researchers will be the experimental groups. The researchers use the resources as their data to come up with the solution to the problem they wanted to enhance such as having an incomplete list. With this, the researchers find the experimental research design fit for this study. This will allow them to manage or control the whole process.

4.1. *Adding descriptive lists to the existing Irving's Algorithm*

Let a = [list of qualities pi is looking for in q where $a_3 < a_1$]
Let b = [list of qualities in q where $b_3 < b_1$]

The descriptive lists describe what each participant is looking for and what they have. By adding descriptive lists, such as "qualities in a roommate," incomplete preference lists can be filled in. These qualities are strictly ordered from most prioritized to least prioritized. The reason they must be strictly ordered is because the most prioritized quality would have the heaviest weight. Additionally, b and a must be the same length.

4.2. Finding the weighted average of an agent using descriptive lists

The closer the qualities of the agents to the order of the qualities an unmatched agent is looking for, the higher the weighted average. After getting all the weighted averages based on the unmatched agent's preference list, the weighted averages are sorted from greatest to least. If the first agent in the newly sorted list is not in the unmatched agent's incomplete preference list, then the first agent will be added to the unmatched agent's list, then move on to the next agent. Otherwise, the list will move to the next agent in the newly sorted list until it reaches the nth agent.

4.2.1. Pseudocode

```
for all qualities in b:
        k = quality in b
        s = arbitrary integer
        if ki exists in a:
                if index of ki is equal to index of b1:
                        weight of ki = s / (s* length of b) * 0.45
                if index of ki is equal to index of b2:
                        weight of ki = s / (s* length of b) * 0.35
                if index of ki is equal to index of b3:
                        weight of ki = s / (s* length of b) * 0.2
                t = t + weight of ki
weighted average of q = t
```

5. Results

5.1. Results in Existing Algorithm

```
INITIALIZATION
prefList
['A', 'b', 'd', 'f', 'c', 'e']
['B', 'd', 'e', 'f', 'a', 'c']
['C', 'd', 'e', 'f', 'a', 'b']
['D', 'f', 'c', 'a', 'e', 'b']
['E', 'f', 'c', 'd', 'b', 'a']
['F', 'a', 'b', 'd', 'c', 'e']
```

Fig. 1. The complete ordered list provided by the proposers

```
prefList
['A', 'b', 'd', 'f', 'c', 'e']
['B', 'e', 'f', 'a', 'c']
['C', 'd', 'e', 'f', 'a', 'b']
['D', 'f', 'c', 'a', 'e']
['E', 'c', 'd', 'b', 'a']
['F', 'a', 'b', 'd', 'c']
```

Fig. 2. Reduced list of each proposer after being rejected

```
prefList
['A', 'b', 'f']
['B', 'e', 'f', 'a']
['C', 'd', 'e']
['D', 'f', 'c']
['E', 'c', 'b']
['F', 'a', 'b', 'd']
```

Fig. 3. Further reduced list after rejecting other participants that are lower than the proposal that is being currently held by the proposer

```
prefList
['A', 'b', 'f']
['B', 'e', 'f', 'a']
['C', 'd', 'e']
['D', 'f', 'c']
['E', 'c', 'b']
['F', 'a', 'b', 'd']
```

```
prefList
['A', 'f']
['B', 'e', 'f']
['C', 'd']
['D', 'c']
['E', 'b']
['F', 'a', 'b']
```

```
prefList
['A', 'f']
['B', 'e']
['C', 'd']
['D', 'c']
['E', 'b']
['F', 'a']
```

Fig. 4. (a) starting from the left, the image shows a list that has > 2 agents; (b) the existing algorithm finds the cycles and reduces the list; (c) after reducing the list, a list of stable matched remain

In the existing algorithm, a preference list is asked from each agent. Uppercase letters denote the proposer or the one who created the preference list, and the lowercase letters denote the items in the list as shown in Fig.1. The characters are used as placeholders for names. Fig. 2 and Fig. 3 shows the list being reduced when the agents reject each other and when the item in the list has lower precedence than the proposal the agent is currently holding. Lastly, in Fig.4, since there are more than 2 items on the list, meaning that a stable matching has not yet been reached, the existing algorithm tries to find the cycles and deletes them accordingly. As shown in Fig. 4 (c), a stable match has been found.

```
['A', 'b','d','f',],
['B', 'd','e','f','a','c'],
['C', 'd','e','f','a','b'],
['D', 'f','c',],
['E', 'f','c','d','b','a'],
['F', 'a','b','d','c',],
```

Fig. 5. List if each agent's preferences with a partially ordered list or incomplete list

```
    if list[0] == proposee.upper():
IndexError: list index out of range
```

Fig. 6. The result of the original Irving's algorithm having an incomplete list

Fig. 5 shows the incomplete list of each agent's preferences. The researchers use single characters as an example for each agent. The uppercase letters are the proposers while the lowercase letters are their preferred agents. The first lowercase letter from the left is the agent that is the highest in the ordered list while the n^{th} lowercase letter is the lowest. With this, as a result seen in Fig. 6, the incomplete list is not accepted in the original Irving's algorithm. This resulted in error and there is no stable matching achieved.

5.2. Results in the Enhanced Irving's Algorithm

Given the same sample stated in Fig. 5, these are the results regarding the enhanced Irving's algorithm.

```
proposerWithQualities = [ #PROPOSER'S QUALITIES
    ['A', 'friendly','clean','respectful',],
    ['B', 'clean', 'friendly', 'honest'],
    ['C', 'responsible','clean','friendly'],
    ['D', 'respectful','responsible','communicative'],
    ['E', 'clean','responsible','friendly'],
    ['F', 'honest','clean','friendly'],
]
proposerLFQualities = [ #QUALITIES PROPOSER IS LOOKING FOR
    ['A',  'clean', 'friendly', 'honest'],
    ['B', 'respectful','responsible','communicative'],
    ['C', 'clean', 'friendly', 'honest'],
    ['D', 'responsible','clean','friendly'],
    ['E', 'clean','responsible','friendly'],
    ['F', 'friendly','clean','respectful',],
]
```

Fig. 7. List of each agent's qualities and the qualities they are looking for in a match

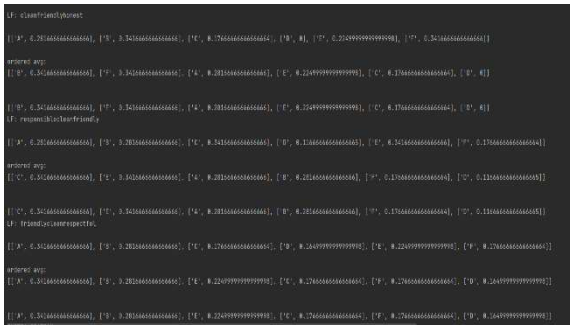

Fig. 8. The enhanced part of Irving's algorithm where qualities are being calculated using weights.

```
prefList
['A', 'b', 'd', 'f']
['B', 'd', 'e', 'f', 'a', 'c']
['C', 'd', 'e', 'f', 'a', 'b']
['D', 'f', 'c']
['E', 'f', 'c', 'd', 'b', 'a']
['F', 'a', 'b', 'd', 'c']


INITIALIZATION
prefList
['A', 'b', 'd', 'f', 'e', 'c']
['B', 'd', 'e', 'f', 'a', 'c']
['C', 'd', 'e', 'f', 'a', 'b']
['D', 'f', 'c', 'e', 'a', 'b']
['E', 'f', 'c', 'd', 'b', 'a']
['F', 'a', 'b', 'd', 'c', 'e']
```

Fig. 9. Filling in the vacancies in an incomplete list using the autocomplete feature of the enhanced algorithm

```
PHASE 1
['A', 'b', 'd', 'f', 'e', 'c']
['B', 'e', 'f', 'a', 'c']
['C', 'd', 'e', 'f', 'a', 'b']
['D', 'f', 'c', 'e', 'a']
['E', 'c', 'd', 'b', 'a']
['F', 'a', 'b', 'd', 'c']
```

Fig. 10. Phase 1 of the Irving's algorithm where agents choose their preferred match

```
PHASE 2
prefList
['A', 'b', 'f']
['B', 'e', 'f', 'a']
['C', 'd', 'e']
['D', 'f', 'c']
['E', 'c', 'b']
['F', 'a', 'b', 'd']
```

Fig. 11. Phase 2 of the Irving's algorithm where list of each agent is being narrowed down

```
prefList
['A', 'b', 'f']
['B', 'e', 'f', 'a']
['C', 'd', 'e']
['D', 'f', 'c']
['E', 'c', 'b']
['F', 'a', 'b', 'd']
```

```
prefList
['A', 'f']
['B', 'e', 'f']
['C', 'd']
['D', 'c']
['E', 'b']
['F', 'a', 'b']
```

```
prefList
['A', 'f']
['B', 'e']
['C', 'd']
['D', 'c']
['E', 'b']
['F', 'a']
```

Fig. 12. Phase 3 (a) the first picture starting from the left shows the list before cycles are searched for; (b) the second picture shows the reduced list while cycles are being deleted; (c) the last picture shows the final matches which turned out to be stable matchings

Fig. 7 shows the list of each agent's qualities, under the variable of proposerWithQualities, as well as the qualities they are looking for in other agents which is under the variable of proposerLFQualities. With the enhanced algorithm, as seen in Fig. 8 and as a part of the initialization process, each quality is being calculated using weights. The quality closest to the proposer's preference will have the heaviest weight and

the farthest quality will have the least weight. Since the qualities now have weighted scores, the enhanced algorithm can now autocomplete the incomplete lists of other agents seen in Fig. 9. The phase 1 of the original algorithm is retained as seen in Fig. 10 where proposers will now send proposals to their preferred matches. Agents who reject each other will be removed from the list and those that remain will move on to Phase 2 as seen in Fig. 11. Phase 2 of the original algorithm is also retained. This phase also narrowed down the proposer's preference list in order to have a match. After Phase 2, Phase 3, as seen in Fig. 12, shows that the cycles that are found are deleted in order to find the stable matchings. As seen in the result, unlike the original algorithm, this enhanced algorithm catered to an incomplete list which is very ideal because not all agents have the same number of preferences. With the help of the autocomplete feature of the enhanced algorithm, it fills up the incomplete list based on the calculation of weighted score on the qualities that gives a stable matching in all agents.

5.3. Comparison with Stable Roommates with Incomplete lists (SRI)

In SRI, the preference list of a participant can consist of less than n-1 members. Once Phase 1 terminates and any participant p ends up with an empty list, although the algorithm will continue, there is a need to consider the cost of an unmatched agent in each stable matching. The value of remaining unmatched or not can be debatable. In comparison with the enhanced algorithm with additional preferences, having an n-1 ordered list that is to the participant's preference would have a higher chance of creating stable matches.

6. Conclusion

After completing the study, the researchers have concluded the following:

- That a complete list of each agent is recommended to have a chance of having more stable matches in an instance.
- That filling the gap of an agent's list close to his/her preference will have a higher chance of having a socially optimal match rather than none.
- That the original algorithm does not cater to incomplete lists.

7. Recommendation

Based on the researchers' findings, conclusions, the researchers recommend the following:

- The researchers recommend the algorithm in matching applications whether it is on the web or mobile devices.
- The researchers also recommend combining other algorithms with this enhanced Irving's algorithm for better operability.
- Future researchers can incorporate machine learning to increase the range of qualities that can be used in the additional preferences and create a closer match to the participants' preferences.

Acknowledgements

The researchers are thankful to God for having bestowed His wisdom and guidance, to our family for accompanying us during the busiest days of our research, to our friends who have supported us in our endeavors, and lastly to the Professors and Staff of Pamantasan ng Lungsod ng Maynila for the support and encouragement that they have provided.

References

Boehmer, N., & Elkind, E. (2020). Stable Roommate Problem with Diversity Preferences *. https://arxiv.org/pdf/2004.14640.pdf?fbclid=IwAR1cp-PZxGo1_l5koNp7W2shEh35TcfpZjfOY7uFEdvLf-tRTEc0t2oJH5U

Chen, Y.-C., & Hu, G. (2018). A Theory of Stability in Matching with Incomplete Information. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.3384439

Cseh, Á., Irving, R. W., & Manlove, D. F. (2017, September). The stable roommate's problem with short lists. www.econstor.eu. http://hdl.handle.net/10419/190487

Irving, R. W., & Manlove, D. F. (2002). The Stable Roommates Problem with Ties. Journal of Algorithms, 43(1), 85–105. https://doi.org/10.1006/jagm.2002.1219

Irving, R. W. (1985). An efficient algorithm for the “stable roommates” problem. Journal of Algorithms, 6(4), 577–595. https://doi.org/10.1016/0196-6774(85)90033-1

Shapely, L., & Roth, A. (2012). Stable matching: Theory, evidence, and practical design. https://www.nobelprize.org/uploads/2018/06/popular-economicsciences2012.pdf

Szestopalow, M. (2010). Properties of Stable Matchings. https://uwspace.uwaterloo.ca/bitstream/handle/10012/5667/Szestopalow_Michael.pdf?sequence=1&isAllowed=y

Yue, D., & Hickman, P. (2016, May 3). CS 136 Final Project: Roommate Matching.Www.academia.edu. https://www.academia.edu/25520849/CS_136_Final_Project_Roommate_Matching?fbclid=IwAR1szEW3qvOYfsFgkKi6VZmEVn4miQg4w2oh2n-Jnc8YmpNXmnxXRnxfIG0