Stable Matching Between Tenants and Landlords in the Rental Market

Ian Paulino
Graduate School of Arts and Sciences
Fordham University, U.S.A.
ipaulinovelez@fordham.edu

***Abstract*—To ease the social constraints of the New York City rental market, a tenant-landlord stable matching algorithm is implemented. A preference-ranking system is developed to create lists of tenants and landlords that comply with the needs of every individual, where the individual that matches with the most preferences of the subject is first in its list, and the least desired is last. These lists are used as input for the Gale-Shapley stable matching algorithm to generate our final pairings. Some equity implications based on the tenant-optimal implementation are discussed, as well as some potential variations and the time complexity of the algorithm.**

Keywords—stable matching, Gale-Shapley, tenants, landlords, rental market.

I. Introduction and Background

To contextualize this work, a brief background overview is to be discussed. While the New York City rental market, renowned as one of the most competitive, is used as an example, the implementation developed in this work can be applied to any similar context. The final objective is to match tenants and landlords to achieve a smoother rental market with less socio-economic consequences using the Gale-Shapley Algorithm and the ranking of individuals based on preferences.

A. Rental Market in New York City

The New York rental market is known for being expensive and competitive, with high demand for rental properties and limited supply leading to relatively high rent prices. One of the main factors contributing to the high demand for rental properties in the city is its strong economy and job market. The city is home to many large corporations, financial institutions, and media companies, which attract a significant number of workers from across the country and around the world. Additionally, New York City is a major cultural and artistic center, with a vibrant arts scene and numerous universities and colleges, which also contribute to the demand for rental housing.

The city has a high population density and a relatively small land area, which means there is limited space for new construction. Zoning regulations, building codes, and other regulatory issues can also make it difficult to build new rental properties in the city. As a result, many people are forced to compete for a limited number of available units, which can lead to high rent prices and bidding wars for desirable properties.

Besides the population dynamics, the rental and housing markets across the world can be highly susceptible to the business cycle, economic bubbles, and an array of other shocks. All these outside forces intervene with a smoother housing process for tenants. The price surge and market tightening create an unbalanced power dynamic, putting the tenants at a disadvantage [1]. Improvements in the rental market must be made to reduce the costs, time spent, and landlord-tenant conflict produced by this unbalance.

The field of Data Science offers solutions and aid to the rental market in all sorts of ways. Different methodologies have been implemented in house pricing analysis, valuation, time series forecasting, among others [2][3]. However, smoothening the relationship between landlord and tenant *during* house hunting to relax the overall rental market has barely been studied in the field.

B. Gale-Shapley Stable Matching Algorithm

The stable matching problem is first studied by Gale and Shapley [4]. The authors wrote about a setting in which two parties exist. Each member in a party wants to match with one (or a higher, fixed number, in some cases) member of the opposite group. The matches have to be done in such a way that no match is deemed "unstable". Gale and Shapley defined unstable by a match in which both members have other two members that prefer them over their current matches. Unstable matches, in most proposed scenarios, will lead to dissatisfaction with their matched partner, knowing that any part of the match could switch to a match that they hold more preference for and would also reciprocate such preference to some level.

The Gale-Shapley Algorithm solves the stable matching problem and creates stable matches across all pairs, albeit with an implication in optimality. Let us say a list of men is to be matched with a list of women in marriage. Each man has a list of preferences for the women in the other list, ranking each individual woman by which they prefer the most, and vice versa. Each man then proposes to the woman of their preference, and unless that woman is proposed to by another man she prefers more, she will stick with that match. This creates stable marriage across all individuals. It is visible, however, that the algorithm will always result in a proposer-optimal solution, yielding the most favorable stable matches for the ones who propose, and the worst for the ones that are asked.

The algorithm has been applied to various real-world stable matching problems. Its own creators first blueprinted it to create stable matches between colleges and college applicants, as well as the aforementioned marriage problem. In [5], one of its most-used applications came to light: medical residency matching, which has evolved the base algorithm in many ways to achieve

new ways to match medical graduates and residency positions. Other modern approaches include matching riders and drivers in ride-sharing systems such as Lyft or Uber [6]. Efforts have been made to match landlords to refugees in Sweden, although no individual preferred any other option other than their current match [7]. Although the method has been applied substantially, tenant-landlord stable matching systems are still relatively uncommon [8].

II. Methodology

The proposed methodology for the tenant-landlord matching consists in the following:

- Create a list of landlords and tenants
- Choose which variables to use for preferences and features in the system
- Collect their preferences and features
- Apply Preference-Ranking Algorithm
- Apply Gale-Shapley Algorithm, with tenants

Each tenant will be ultimately matched with one unique landlord. For simplicity in this work, the words *landlord* and *apartment or house* are used interchangeably. If one landlord were to have two apartments for rent, the apartments would be matched separately each with one different tenant as if it were two landlords in the system.

A. Lists of landlords and tenants

Two lists, one of landlords and one of tenants is created. Each list would contain objects with certain characteristic features and certain preferences. This could be based on surveys or questionnaires for the purpose of real-world application.

```
class tenant:
  initialization:
    features ← name, index, income, credit_score, rental_history,
    guarantor, references, pets
  ranking ← {}
  representation:
    return name
  def add_preferences(sq_feet, rent, location, bedrooms, bathrooms,
  security, allows_pets, laundry, parking, maintenance):
    preferences ← [sq_feet, rent, location, bedrooms, bathrooms,
          security, allows_pets, laundry, parking, maintenance]
class landlord:
  initialize:
    features ← name, index, sq_feet, rent, location, bedrooms
  bathrooms, security, allows_pets, laundry, parking, maintenance
  ranking ← {}
  representation:
    return name
  def add_preferences(income, credit_score, rental_history, guarantor,
  references, pets):
    preferences ← [income, credit_score, rental_history, guarantor,
    references, pets]
```

The most important part in this step is that the members of each class are set in such a way that they hold the features their counterparts have under preferences. Tenants would initialize with features of income, credit score, and whether they have good rental history, a guarantor, references and pets, while landlords have all these features as qualities they will have a preference for, so that they can eventually be matched. Similarly, landlords have tenants' preferences as attributes, which are amount of square feet, monthly rent, location, amount of bedrooms and bathrooms, and whether they have security facilities, parking, maintenance, laundry facilities and allow pets.

B. Preference-Ranking Algorithm

In the next step, each tenant and landlord will rank each member of the other party based on their preferences. The algorithm will first iterate for each tenant over each landlord, placing a score on each for that tenant. The algorithm will check whether for that tenant and a landlord, their list of preferences and features, respectively, match. For example, if a tenant prefers square feet of more than 800, an apartment in Queens, two bedrooms and two bathrooms, the algorithm will assign a score of +1 for each of these preferences that match with the landlords' attributes. The tenants will end up: first, with a list of rankings for each landlord (integers). Then, the list of landlords will be sorted based on their score in descending order, generating a list of sorted landlord names for each tenant, in the order of their preference.

```
Input: tenant_list, landlord_list
Output: tenant_preferred_set_sorted, landlord_preferred_set_sorted
for t in tenant_list:
  for l in landlord_list:
    score ← 0
  if tenant.preferences[sq.feet] <= landlord.sq_feet:
    score ← score + 1
  if tenant.preferences[rent] >= landlord.rent:
    score ← score + 1
  for i = 2 to length(tenant.preferences):
    if tenant.preferences[i] == landlord.features[i]:
      score ← score + 1
  tenant.ranking[landlord] = score
sort(tenant.ranking) by value
```

The same process will be repeated for landlords, with the caveat that some features in this trial have a conditional relevance to other features' existence. For example, a landlord will most likely require a guarantor, only if the tenant does not meet the credit score and/or income requirements. A tenant that both meets the requirements and has a guarantor will have an overestimated score, while one that has only the guarantor may have an underestimated score, considering that sole guarantor might be able to override the relevancy of having good credit

score or income. This is dealt with during this part of the algorithm:

```
for l in landlord_list:
  for t in tenant_list:
    score ← 0
  if landlord.preferences[income] <= tenant.income and
  landlord.preferences[credit_score] <= tenant.credit_score:
    score ← score + 2
  else if tenant.guarantor == True:
    score ← score + 2
  for i = 2 to len(landlord.preferences):
    if landlord.preferences[i] == tenant.features[i]:
      score ← score + 1
  landlord.ranking[tenant] = score
sort(landlord.ranking)by value
```

C. Gale-Shapley Algorithm

Finally, using the sorted preference set of each tenant and landlord, we can apply Gale-Shapley's stable matching algorithm to generate our stable matches. The tenants are the proposers. Each tenant will choose the first landlord on their list (their most preferred) to match with, and a match will be created. If another tenant prefers that landlord over the other available ones, and that matched landlord prefers the new tenant over their current match, a new match will be created between the two, leaving the newly unmatched tenant to choose the next landlord on their list. This loop will be repeated until no landlord or tenant is able to change to a match they have more preference for.

```
Input: tenant_preferred_set_sorted, landlord_preferred_set_sorted
Output: Stable matches
initialize:
  MATCHES ← [ ]
while some tenant t is unmatched and hasn't proposed to
every landlord:
  l ← first landlord on t's list to whom t hasn't proposed to yet
  if l is unmatched:
    append t − l match to MATCHES
  else if l prefers t over current match:
    replace t − l
  else:
    l rejects t
```

III. Discussion

There are some important takeaways to be discussed about this implementation in the rental market. This algorithm, besides working with any list of *n* landlords and *n* tenants, has an equity implication in terms of the optimality. However, this specific example has some limitations that could be overcome in future, branching works. Finally, the time complexity of the overall system will be brought to light.

A. Optimality

As we've discussed, the Gale-Shapley algorithm is always proposer-optimal. That said, it favors whoever "proposes" in the resulting stable matches, ultimately linking the proposers with their most preferred options, while the ones "proposed to" end up with relatively worse matches. This is no exception to the rule.

Our tenant-landlord matching algorithm puts the tenants as the proposers. Since the tenants match with their preferred landlords first, the algorithm is tenant-optimal. In a real-world application, this would mean the tenants end up with the better side of the deal, and thus hold more sway in the power dynamic than if it were one landlord choosing from a pool of tenant candidates, as well as easing the competition among renters.

B. Limitations and variations

This specific implementation has some useful variations that need to be kept in mind. When choosing the variables for our algorithm, a multitude of different parameters could be chosen among tenant and landlord features and preferences. Besides from the list presented in this work, the floor in which an apartment is located, whether it has a balcony, backyard or common area, whether smoking is permitted, and the payment method and deadlines landlords ask for may be taken into account. Any kind of criteria that tenant and landlords often look for may be fit into the preference-ranking algorithm with minimal effort and effective planning beforehand.

It is also useful to denote the specific formula used in this preference-ranking algorithm. The process assigns a score of 1 more for each matching criteria between the landlord and the tenant, but more useful and context-relevant weighting can be implemented. It might be useful to rank good credit score and income higher with a higher score for landlord preferences or assign a higher score to landlords with the right location in tenants' lists in a general sense. It might also be useful to go even deeper with different score weighting for each individual landlord and tenant, allowing them to hold a final preference list more akin to what they are actually looking for.

Finally, the real-world implementation of the system, while completely viable, might be faced with some rejection from landlords. Thus, the application has to be smart and incentivize landlords to participate with minimal loss.

C. Time Complexity

The preference-ranking algorithm has nested loops involving all tenants over all landlords and vice versa. This part runs in $O(n^2)$, where n is the number of tenants/landlords (same number of tenants and landlords is required). While a nested loop that goes over the preferences and features and matches accordingly exists, the preferences will often hold a minimal number compared to the input amount of tenants and landlords themselves.

These lists are then sorted. The time complexity of whichever sorting mechanism is used should be considered. Lastly, the Gale-Shapley Stable Matching runs in $O(n^2)$, since it iterates each tenant over each landlord in the worst case to get all matches.

IV. Conclusion

To ease the power balance created by the hike of prices and tightening of the rental market in New York City, the stable-matching problem between a list of tenants and landlords was solved. Each tenant and landlord offer a list of preferences for the other, and a preference-ranking algorithm was developed to assign scores to each individual for each counterpart based on the number of requirements they comply with. A final sorted list of all individuals in the counterparty was made for each individual, with the individual they prefer the most in first place, and the least preferred in the last place.

This list is used as input for the Gale-Shapley algorithm, and the stable matches are created in an overall runtime of $O(n^2)$. The algorithm was chosen to be tenant-optimal to conform to the equity and social implications of the real-world rental market. This algorithm can be improved in various ways in future works, such as with better weighting and specification of the preference scores, or with a larger number of preference variables to consider.

References

[1] Satsangi, M., & Kearns, A. (1992). The use and interpretation of tenant satisfaction surveys in British social housing. Environment and Planning C: Government and Policy, 10(3), 317-331.

[2] Kou, J. (2022). Analysing Housing Price in Australia with Data Science Methods (Doctoral dissertation, Victoria University).

[3] Afonso, B., Melo, L., Oliveira, W., Sousa, S., & Berton, L. (2019, October). Housing prices prediction with a deep learning and random forest ensemble. In Anais do XVI Encontro Nacional de Inteligência Artificial e Computacional (pp. 389-400). SBC.

[4] Gale, D., & Shapley, L. S. (1962). College admissions and the stability of marriage. The American Mathematical Monthly, 69(1), 9-15.

[5] Gale, D., & Sotomayor, M. (1985). Some remarks on the stable matching problem. Discrete Applied Mathematics, 11(3), 223-232.

[6] Wang, X., Agatz, N., & Erera, A. (2018). Stable matching for dynamic ride-sharing systems. Transportation Science, 52(4), 850-867.

[7] Andersson, T., & Ehlers, L. (2020). Assigning refugees to landlords in Sweden: Efficient, stable, and maximum matchings. The Scandinavian Journal of Economics, 122(3), 937-965.

[8] Bristi, W. R., Chowdhury, F., & Sharmin, S. (2019, July). Stable matching between house owner and tenant for developing countries. In 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.