

OTDK-dolgozat

2025

Corvinus University of Budapest

Az optimális párválasztási stratégia ágens-alapú modellezése

Agent-based model for exploring optimal dating strategies

Szerző: Mirkó Tibor Szakali

Konzulens: Mánuel László Mágó

Kézirat lezárásának dátuma: 2024. 04. 10.

Contents

1	Introduction	1
2	Literature	4
3	Methodology	6
3.1	Creating the Agents	6
3.2	Creating the Simulation	7
3.3	Principal Component Analysis	10
3.4	Sensitivity Analysis	11
4	Results	12
5	Discussion	18
	References	23
	Appendix	24
5.1	Pseudocodes	24

List of Figures

1	Correlation Heat Map Between Q-Values and Agent Attributes	13
2	PCA of Agent Decisions Under Sufficient Temperature Parameter	14
3	PCA of Agent Decisions Under Low Temperature Parameter	15
4	PCA of Agent Decisions With 3 Clusters	17

Abstract

I formulate and simulate an agent-based model to explore optimal strategies within a dynamic dating market. The model aims to replicate real-world interactions through agents' attributes, preferences, utilities, and behaviors, as grounded in relevant literature. The agents have incomplete information about their own preferences and the attributes of others. They learn to adjust and refine their own preferences over time based on the feedback (utility) received from interactions. The interactions happen in a 2-dimensional space, the incorporation of the spatial element simulates proximity-based preferences and homophily. With the use of reinforcement learning (RL) algorithms I simulate agent behaviors and evaluate their performance using the complete information stable equilibrium point (Gale-Shapley) as benchmark. This evaluation reveals optimal stopping points where the expected utility of maintaining a current relationship exceeds potential benefits derived from new encounters. I find that attractive people have significantly shorter waiting times before securing partners, a phenomenon amplified by the network effects associated with the ratio of attractive individuals in the population. My model achieves, on average 71.1% of the welfare realized by the Gale-Shapley algorithm. However, the personality traits do not have predictive power regarding dating strategies in my model. These findings contribute to understanding strategic behavior not just in dating markets, but also offer implications for the economic theory of two-sided markets.

1 Introduction

In order to find patterns in such an intricate topic as dating, one must refrain from using oversimplified analytical models but also avoid excessive variables. For this reason, I built my own agent-based model¹ in Python without using external libraries.

In my simulation, each agent is assigned attractiveness scores and personality scores, along with preference weights for their potential partners. Although all agents prefer more attractive partners, their rankings vary because they value different attractiveness characteristics, as determined by their preference weights. Personality-wise, I make the assumption of homophily: agents prefer those who are similar to themselves in terms of inner characteristics. Conventionally attractive agents also have an absolute advantage in the dating market as they are able to meet more people at each time step of the simulation. To avoid random and unsatisfying interactions, I implemented 7 different locations that agents can visit. This feature supports assortative matching and also imitates subcultures that tend to form in real-life.

At the beginning of the simulation there is an exploration phase, when agents form their perception of an ideal partner. After enough interactions have happened, the exploration phase ends, and agents actively try to ‘stop’ or date when their partner exceeds a certain threshold set by their best partner from the exploration phase. We can refer to this threshold value as their ‘standards’ for a relationship. Note that selecting the best partner in the exploration phase as a benchmark is analogous to the optimal strategy (Odds algorithm) in the well-known Secretary Problem that has roughly a length of $\frac{n}{e}$ (n being the number of expected applicants, e referring to Euler’s number) rejection phase and optimally chooses the first applicant that exceeds the quality of the best applicant from the rejection phase. However, the dating problem cannot be solved using the Odds algorithm, as it requires both parties to agree to start dating. So we can refer to the problem of dating as a two-sided optimal stopping problem. The two-sided version of the Secretary Problem has no definite solution. Analytical solutions exist only if we can make assumptions on how cooperative the other agents are, see Eriksson et al. (2007), and we have no reason to make such assumptions in a dating setup.

It is a realistic assumption that people have a tendency to lower their dating standards over time, if they are unable to find a partner. The agents make decisions about how

¹An agent-based model is a stochastic simulation in which agents interact and make decisions.

much they are willing to lower their minimum requirement of a partner. The optimization problem becomes exceedingly difficult due to the simulation's dynamic nature. As the pairs of agents who managed to find a date are removed from the simulation (reaching their terminal state), agents are not only faced with the dilemma of over-exploring or settling too early, they also have to take into account the constantly changing dating pool.

The decision making process was implemented with the use of Q -Learning. But as the agents have very limited time to learn and because of the dynamic nature of the problem any real learning can only take place if we allow the agents to 'relive' their dating lives and start over again. This way we can see how well the agents can learn the game they are playing.

The agents can be characterized as boundedly rational, as they do not try to find their absolute best match. If that was the case, my model would be analogous with a stable matching algorithm, meaning that agents would evaluate all their possible partners before settling. Instead, in my simulation the agents' perception of their ideal partner is based on their experiences in the exploration phase of the simulation. Agents try to maximize their expected payoff under the assumption that their reward would be sufficient when settling with the best candidate from the exploration phase. So in my model agents search for a 'good enough' partner instead of a best partner.

The simulation framework developed in this study provides a valuable tool for analyzing the dynamics and equity of two-sided dating markets. By enabling the adjustment of gender ratios and the clustering of agents based on personality and attractiveness, the simulation allows for the examination of network effects from both cross-side and same-side perspectives. The flexibility in manipulating group sizes offers a foundation to explore the impact of various market structures on the efficiency and fairness of matchmaking outcomes.

The goal of this paper is to answer the question 'Is there a definitive optimal strategy for dating or is it completely arbitrary?' and also to provide a framework for a more sophisticated approach to simulate dating environments, to seek for emerging patterns and see if those patterns align with the economic or psychological literature, especially the theory of two-sided markets. Understanding such a complex mechanism as dating on a societal level would provide valuable insight for companies for targeted ads or dating rec-

ommendations. Evidently, people with different characteristics will have different optimal dating strategies and if we could predict these patterns based on empirical data, we could enhance cross-side network effects by connecting the right people on a dating platform.

Consistent with what might be intuitively understood, the simulation’s findings suggest that individuals perceived as less attractive endure longer waiting periods before securing a partner, illustrating the significant influence of physical attractiveness on dating success within the framework of this model. Interestingly, no personality trait emerged as a reliable predictor of dating success. A particularly notable observation was the distinct advantage enjoyed by males with the highest attractiveness scores, who demonstrated significantly shorter waiting times before forming partnerships. Moreover, the competitive dynamics within the dating pool further magnify this effect; attractive males, being a comparatively rarer demographic, face less competition. This relative scarcity enhances their standing within the simulated dating market, thus providing them with a substantial advantage in securing partners swiftly.

The paper is organized as follows: Section 2 provides an overview of the various interdisciplinary literature used in my model. Section 3 summarizes the creation of my model and the analysis step by step. Section 4 illustrates the results, and Section 5 provides my interpretation of the model, final remarks, and ideas for further research.

2 Literature

My work is built upon various assumptions that are drawn from interdisciplinary literature. De La Mare and Lee (2023) and Neyt et al. (2020) have found that on dating applications people generally prefer a partner with similar personality traits. Following these research papers I developed assortative mating in my simulation. I have also implemented subjectivity with regards to attractiveness as (Ibáñez-Berganza et al., 2019) find that given freedom to modify facial attributes to create an ideal face, subjects constructed different faces which aligns perfectly with the theory that attractiveness is by a large part subjective.

The Gale-Shapley stable matching algorithm (Gale and Shapley, 1962) provides a Pareto efficient solution to the dating problem. It serves as a great benchmark for comparing outcomes on the well-being of the whole agent population. However, it is noteworthy that the Gale-Shapley algorithm favors those who make propositions and is less desirable for those who receive these propositions. For this reason we must compare the outcome of my simulation with a both male- and female optimal stable matching.

With regards to agent-based simulation approach to dating, Shaheen et al. (2022) have simulated an online dating platform to enhance optimal policy-generation process of online social networks. How one constructs the social network will fundamentally affect the outcomes of the model. The formation of social networks becomes relevant, because I integrated a spatial element in my model, as agents can choose their locations and only interact with the people at the same location. If agents only visit the same location over many iterations that implies that many isolated subcultures have formed in the population. Following the work of Centola (2015), homophily and consolidation are the two driving factors that determine how isolated different groups will become. Consolidation refers to the correlation between a person’s characteristics. If consolidation is high, then agents form groups based on certain characteristics, such as personality traits. These traits are good predictors of all other characteristics of potential partners. In my model, while homophily is present, there is no correlation between personality traits and attractiveness; therefore, we can expect an intersecting social network with high mobility from one place to another.

As highlighted by (Ligmann-Zielinska et al., 2020) it is important to categorize agent-based models into either abstract theoretical models (that aim to discover fundamental

principles) or application driven agent-based models (which are built in order to explain a specific phenomenon). My simulation can be put in the former category as I lay out a general framework which can be later tested under specific irregular circumstances.

The implementation of the optimal stopping problem is widely used in finance to determine the price of American options. A generalization of the Black-Scholes model is needed as American options can be exercised at any time before expiration (Jacka, 1991). I modeled the problem as a Markov Decision Process (MDP) and I use Q -Learning to determine the optimal stopping points. I try to discretely approximate the Q -Function for optimal stopping with Q -values, a simplified version of the research of Ery and Michel (2021) and Becker et al. (2019).

The dating market can be viewed as a two-sided market, where one of the primary factors that determine the success of a dating platform is the size of the market, as users demand economies of scale to obtain greater variety and choice of potential partners. Another important factor is the network effects of the platform. Contrary to intuition, the gender representation is optimally not equal from the perspective of firms providing dating services. Voigt and Hinz (2015) show that dating platforms maximize revenue when their users are only 36.2% female. This is due to the fact that 'women exert stronger positive cross-side network effects on men than vice versa'. This research suggests that asymmetric network effects are present in modern dating applications. This serves as an interesting addition to my findings.

As far as I know, my model is unprecedented as agent-based models have not yet been used to model a dynamic and complex dating environment.

3 Methodology

3.1 Creating the Agents

I represent agents with numeric values in both inner and outer characteristics for the sake of efficient computation and simplification. Agents are given different attributes such as attractiveness scores and inner qualities. These qualities both range from 1-10 but they serve a different purpose. The attractiveness of a person will boost the overall success of attracting partners, while the inner qualities serve to estimate the similarity of agents' personalities. The attractiveness scores are on an interval scale 1 being the worst and 10 the best. The objective attractiveness is calculated by simply summing the values of the attractiveness vector. The inner qualities of an agent give the spectrum of their personality traits. It is common in the field of psychology to measure personality in such manner, for example The Big Five Personality Test (Digman, 1990). Being merely a relational variable, it measures only the presence or intensity of a given trait. The attractiveness and personality scores follow a normal distribution that differ in standard deviations and in the case of females, I increased the mean attractiveness score, assuming there are fewer attractive males in my simulated society, to study network effects. Each agent was given weights which determine how important the agent finds a personality characteristic or an attribute related to attractiveness. These preferences are arbitrary in real life, so each agent was assigned a random weight drawn from a uniform probability distribution. These weights ensure that the agents have subjective preferences, and it makes the pairing algorithm more realistic. Each agent's attractiveness limits how many people they can interact with in each iteration. This is based on research Ha et al. (2010) that conventionally attractive people have a greater chance to be approached or to be invited on dates. The factors that determine attractiveness, such as eye color, height, facial symmetry, and fitness, were not included in my model because they seem redundant. Instead, I used just a numerical score based on three separate attributes to determine attractiveness.

The most vital part of my study is how I construct the utility function that determines the reward of dating for agents. The utility function U is defined as a combination of the utility received from the partners appearance and the utility of how similar the two agents are personality-wise, where:

$w = [w_1, w_2, \dots, w_n]$ is the vector of appearance weights.

$a = [a_1, a_2, \dots, a_n]$ is the vector representing another person's appearance.

$q = [q_1, q_2, \dots, q_m]$ is the vector of the agent's inner quality preferences.

$q' = [q'_1, q'_2, \dots, q'_m]$ is the vector representing another person's inner qualities.

The utility function U combining both appearance and inner quality utilities can be written as:

$$U(w, a, q, q') = \sum_{i=1}^n w_i a_i - \frac{1}{m} \sum_{j=1}^m (q_j - q'_j)^2.$$

This function calculates the total utility derived from another person's appearance and inner qualities, based on the agent's preferences. The utility from appearance is calculated by taking the dot product of the appearance weights vector and the other person's appearance vector. The utility from inner qualities is calculated by averaging the squared differences between each corresponding element of the agent's inner quality preferences vector and the other person's inner qualities vector. This value is then subtracted from the appearance utility.

3.2 Creating the Simulation

The simulation lasts 50 time periods. In each iteration, the agents choose a location by a simple epsilon-greedy algorithm², meaning that they select the location where they anticipate the highest cumulative utility, based on past experiences.

When creating interactions, we have to eliminate any discrepancies resulting from agents not meeting in the correct order. Evidently, if an agent(i) meets agent(j) at a given index of its interaction vector, agent(j) must meet agent(i) at the same index in his own interaction vector, and no other agent can meet either of them at that index during the iteration. Mathematically, this can be formulated as follows: Consider the sets of male and female agents, $X = \{x_1, x_2, \dots, x_n\}$ and $Y = \{y_1, y_2, \dots, y_m\}$, respectively. We introduce a function $f : X \times Y \rightarrow \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers. This function assigns an interaction index to every pair of male and female agents, such that for $\forall i$, $f(x_i, y_j) \neq f(x_i, y_k)$ where $j \neq k$ and $\forall j$, $f(x_i, y_j) \neq f(x_k, y_j)$ where $i \neq k$.

²The epsilon-greedy algorithm is utilized to balance the trade-off between exploring new locations and exploiting previously visited locations that yielded high utility. This ensures that while agents often choose the best-known option, they occasionally explore less-known options to discover potentially better solutions.

This requirement is satisfied in Python using randomized Latin squares of the interaction vectors. Afterwards, the vectors are transformed in order to match each person’s interaction limit set by attractiveness.

The decision making process is about the agents’ choice of at what rate to lower their minimum requirement of a partner. This process was implemented using a Q -Learning algorithm. Q -Learning is a model-free reinforcement learning algorithm that maximizes the expected reward (in this case finding a most suitable partner) see also Sutton and Barto (2018) for reference. The generic Q -Learning equation is given as follows:

$$Q(s, \delta) \leftarrow Q(s, \delta) + \alpha \left[R(s, \delta) + \gamma \max_{\delta'} Q(s', \delta') - Q(s, \delta) \right].$$

Where $Q(s, \delta)$ represents the Q -value for being in state s and choosing the rate of lowering expectations δ . The states along with the actions were developed in a discrete manner. The simulation switches states after every 10 iterations, this creates the opportunity to see how agents act differently in later stages of the simulation. Parameter α is the learning rate, controlling the extent to which the new information will override the old information and γ is the discount factor, indicating that finding a partner early is preferred to finding them later. Note that both the α and γ values can be chosen somewhat arbitrarily, the right values for them depend entirely on how the simulation was set up in the first place. I have tried several values for these variables for the sake of sensitivity testing. $R(s, \delta)$ is the immediate reward received after applying the rate to settle δ in state s . The $\max_{\delta'} Q(s', \delta')$ expression finds the maximum future reward achievable from any possible settle rate δ' in the next state s' . The expression

$$R(s, \delta) + \gamma \max_{\delta'} Q(s', \delta') - Q(s, \delta)$$

calculates the difference between the estimated future rewards and the current Q -value, adjusted by the learning rate α to update the current Q -value.

Thereafter I run the simulation multiple times to see if agent’s would change their behavior if they could re-live their dating life once again. A seed was given for creating agents to ensure that the exact same agents participate in dating in each iteration. The seed ensures reproducibility and to observe *ceteris paribus* effects of changes in the

environment. This gives us the following formula, now extended with policies:

$$Q^\pi(s, a) = \sum_{s', r} P(s', r|s, a) \left[r + \gamma \sum_{a'} \pi(a'|s') Q^\pi(s', a') \right].$$

The action-value function $Q^\pi(s, a)$ represents the expected return (or total discounted rewards) for taking an action a in a state s and thereafter following a policy π . The policy can be thought of as the combination of the agent's belief about their potential partners and the agent's optimal decision of when to settle with those partners. $P(s', r|s, a)$ denotes the probability of transitioning to state s' and receiving reward r after taking action a in state s . (That is the probability that the agent's proposal to date will be rejected). Parameter γ is the discount factor that balances the importance of immediate versus future rewards, and $\pi(a'|s')$ is the probability of taking action a' in the next state s' under policy π . This function iteratively calculates the value of each action in each state.

However, with Q -Learning I ran into the issue. In the early stages, agents choose risk averse strategies of lowering their standards immediately. This showed that a simple epsilon-greedy strategy of taking actions might not be realistic and lead to false results. To ensure that the level of exploration is sufficient I needed to implement a more sophisticated approach.

The probability of choosing action a at state s is given by the Boltzmann (softmax) exploration formula:

$$P(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a'} e^{\frac{Q(s,a')}{\tau}}}.$$

Where the τ temperature parameter starts with relatively high values and it decays with the number of iterations. This approach ensures more exploration in the early stages of the simulation as with high τ values the probability of each action is close to uniform. As the simulation progresses and the agent gains experience, τ decreases, causing the probabilities to become more skewed towards actions with higher Q -values. This shift allows the agent to exploit the strategies that have proven to be more successful in the past while still maintaining a degree of exploration to avoid local optimum. Note that reason why we want to avoid a local optimum is because it could result in suboptimal dating strategies across the whole simulation. Hypothetically, if a significant portion of the agents prematurely settles for less and lowers their standards substantially, the optimal choice for the remaining agents would be to adopt a similar strategy of lowering their

expectations. This could lead to a panic phenomenon akin to what is observed during financial crises.

My algorithm requires one degree of nesting, as I need to run the simulation multiple times to estimate the Q -Values used in the primary simulation. However, with only one degree of nesting, this nested simulation do not contain further Q -Learning algorithms.

3.3 Principal Component Analysis

After the simulation has run it is time to analyze the data. As each agent can be represented as a 23-dimensional vector, including all their decisions and characteristics, I have used principal component analysis (PCA) to reduce dimensionality and visualize the key results of the simulation. It is common for agent-based models to use this approach, see Camargo (2020). Mathematically, PCA follows these steps:

Given the dataset with n agents and m variables for each agent, X is an $n \times m$ data matrix. We should standardize X to obtain Z , where $Z_{ij} = \frac{X_{ij} - \mu_j}{\sigma_j}$, with μ_j and σ_j being the mean and standard deviation of the j -th variable. Then calculate the covariance matrix S of Z as $S = \frac{1}{n-1} Z^T Z$.

The next step is the Eigen Decomposition: We solve for the eigenvalues λ_i and eigenvectors v_i of S , such that $Sv_i = \lambda_i v_i$ for $i = 1, \dots, m$. Then select the Principal Components: Sort the eigenvalues λ_i and corresponding eigenvectors v_i in descending order of λ_i . Select the first k eigenvectors to form the matrix W , where W is an $m \times k$ matrix and $k < m$.

Lastly, project the standardized data Z onto the new subspace defined by W to obtain the principal component scores Y as $Y = ZW$, where Y is an $n \times k$ matrix representing the data in the reduced principal component space.

With the orthogonal transformation we can convert the set of agents and their decisions into a set of values of linearly uncorrelated variables (Principal Components). This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component, in turn, has the highest variance possible under the constraint that it is orthogonal to the preceding components. With this statistical procedure we can visualize our data. For machine learning applications and a precise algebraic deduction see Goodfellow et al. (2016).

3.4 Sensitivity Analysis

I have done several local and global sensitivity tests³ on my model, adjusting learning rate, discount factors, number of agents and number of iterations. The results were robust across most settings, except under very low τ values, which induced risk-averse behavior among agents. This behavior led to a willingness to lower standards more across iterations, increasing the frequency of pairing but at lower expected rewards, as explained above. I have also adjusted the number of attractive people in both genders to observe the network effects of different agent populations.

After running several configurations of different attractiveness distributions of my agent population, my findings are completely aligned with two-sided market theory. When I ran the simulation with a higher ratio of highly attractive females, it boosted the males' utilities, thus creating a positive cross-side effect. However, this adjustment also induced a negative same-side effect for females, particularly those with lower attractiveness scores. These female agents faced increased competition and were often perceived as less desirable within the simulation environment. This dynamic highlights the challenges of maintaining balance in a two-sided market where one side's gain can inadvertently lead to decreased outcomes for another group within the same side. Interestingly, in scenarios where highly attractive agents were scarce, both male and female, these agents tended to perform poorly. This counterintuitive finding may stem from their limited options and the high competition they face for each other's attention, which can prevent them from forming matches as effectively as in more balanced environments.

³Local sensitivity analysis involves adjusting one parameter at a time to observe the changes in the model output, commonly referred to as a 'one-at-a-time' (OAT) method. In contrast, global sensitivity analysis examines the effect of simultaneously varying multiple parameters, which helps in understanding the interactions and combined effects of all parameters on the output.

4 Results

After running the simulation, the Python code generates an Excel (.xlsx) output file that can be used for analysis. This output contains the Q -Values of each action corresponding to any given state of the simulation. The model considers three possible actions, which determine the willingness to lower standards, and divides time into 5 states. This setup results in 15 Q -Values. Upon analyzing the data, it becomes apparent that each agent has a somewhat predetermined optimal time to date. This is consistent even if the simulation is run 1000 times with the same agents, using a high exploration strategy provided by the Boltzmann exploration function. This means that the optimal waiting time for each agent is not random at all. Thus, each agent has an optimal strategy at an individual level. To see emerging patterns I tried to aggregate agents into categories based on attractiveness and personality traits to see if some given traits correlate with the optimal stopping time. To see if these results are statistically significant, I used the Kruskal-Wallis H Test. I could not use a simple two-sample t-test or Parametric Analysis of Variance (ANOVA) because the Q -Values do not form a normal distribution, so the assumption of normality is violated. Using the Kruskal-Wallis H Test and Welch's t-test (from the 'scipy' library in python) I found that my findings below are statistically significant.

My results show that based on attractiveness, most agents have an optimal waiting time of 100 iterations. Between 100 and 200 iterations the Q -Values are significantly larger, indicating a higher aggregate reward. An exception from this pattern is observed among male agents with high attractiveness scores. The simulation results for this subset of agents suggest that their most advantageous strategy, in terms of average reward maximisation, is to exit the simulated dating market relatively early. This tendency can be attributed to the underlying assumption of our model where female attractiveness is characterised by a normal distribution with a mean value higher than that of males. Consequently, highly attractive males encounter favourable selection dynamics early on, thus reducing their incentive to prolong their market presence. As seen in the illustration of the correlation heat map between agents' attributes on 13, it is clear that for each state (meaning for every 3 subsequent Q -Values) the values are highly correlated in early states of the simulation. However, this correlation fades away as we proceed to later stages. This can be due to the fact that simply less dates happen when most agents have already quit the simulation,

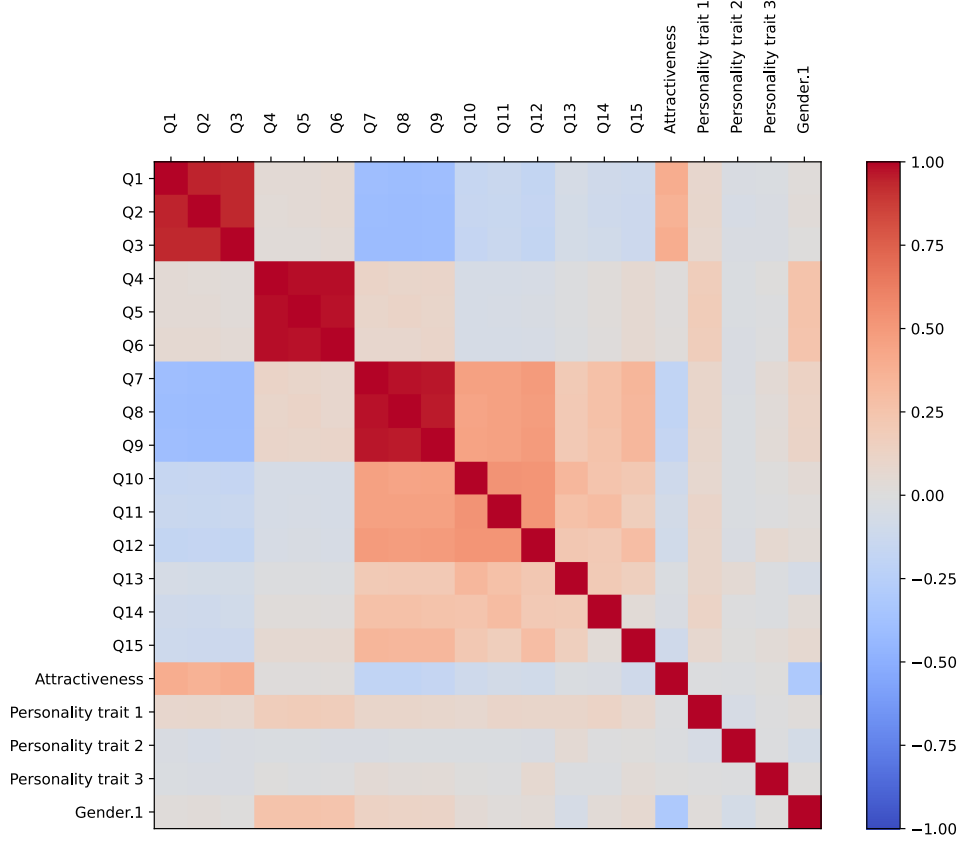


Figure 1: Correlation Heat Map Between Q-Values and Agent Attributes

and the occasional matches are rare and unpredictable.

In the modeled environment, agents with lower attractiveness scores exhibit a higher likelihood of not securing a partner during the initial phases of the simulated dating landscape. These agents are, consequently, required to extend their search over a prolonged duration, systematically reducing their selectiveness to increase their probability of finding a match.

Compared to the Gale-Shapley algorithm, my model produces lower aggregate welfare. This outcome is anticipated to some extent, given that a subset of the agents, particularly those deemed less desirable, remain unmatched. This is due to the fact that agents who are perceived as less desirable would have to lower their standards substantially, otherwise they cannot find a partner. Moreover, in my model agents do not aim to find their absolute best match, instead they will search for a partner with whom they reach a sufficient satisfaction level. As bounded rationality is introduced into the model, the aggregate utility will most likely be lower than the Nash equilibrium point provided by the Stable Matching algorithm.

3-Dimensional PCA of Dating Strategies

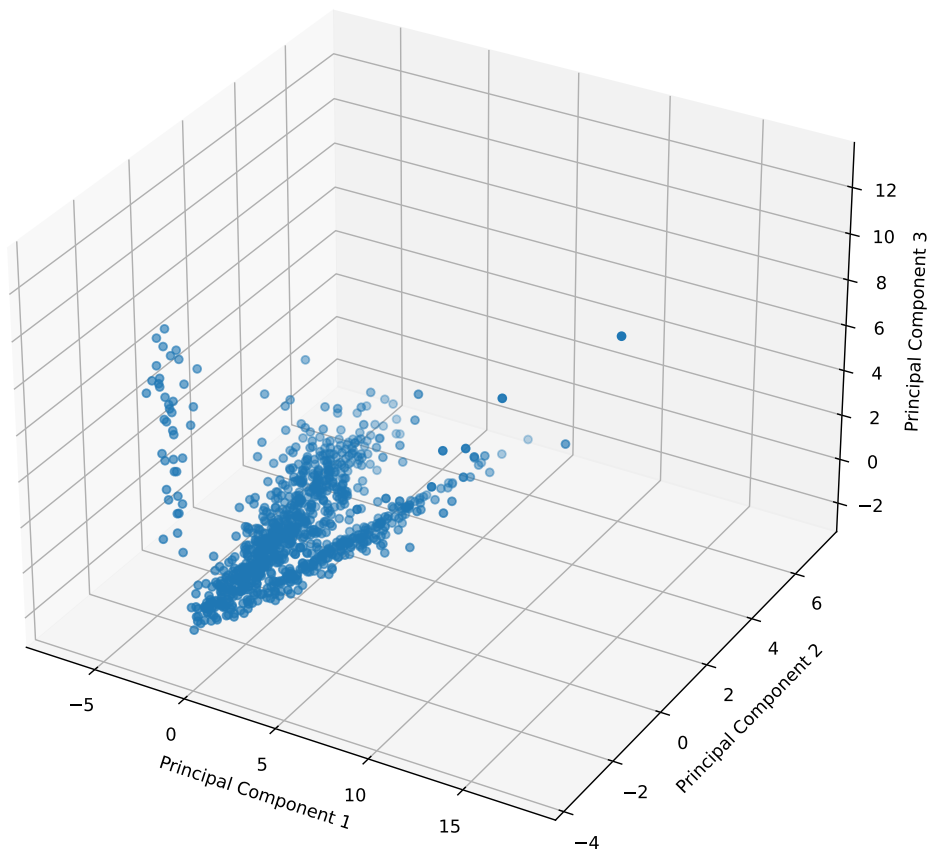


Figure 2: PCA of Agent Decisions Under Sufficient Temperature Parameter

3-Dimensional PCA of Dating Strategies

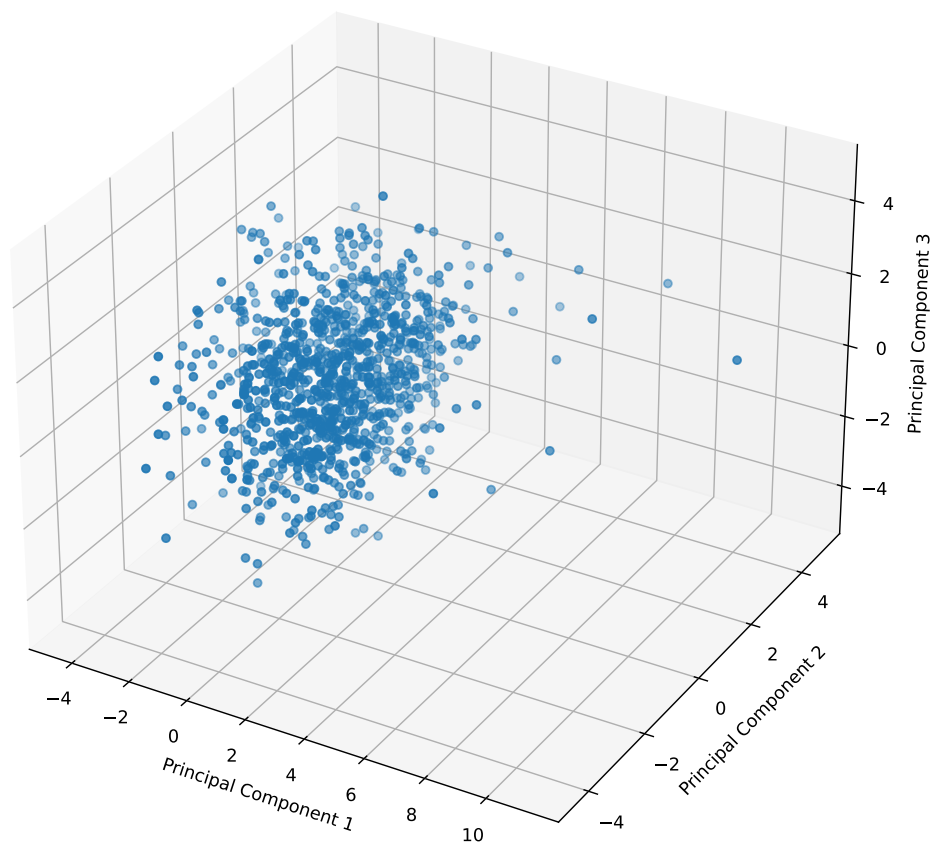


Figure 3: PCA of Agent Decisions Under Low Temperature Parameter

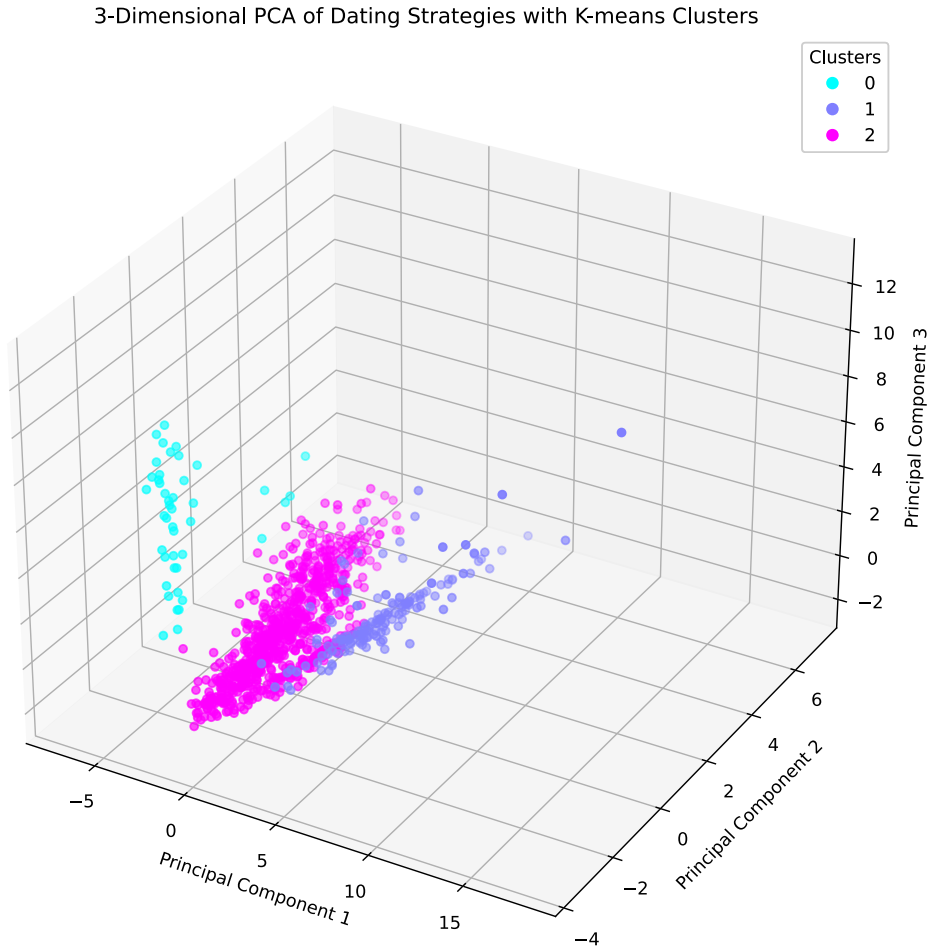
The illustrations Figure 2. and Figure 3. clearly show that one of the most important tasks in machine learning is to set the temperature parameter high enough that agents can explore their options sufficiently. The Figure 3. is the result of a simulation run with a very low τ value of 3. This results in an over-exploitation, where agents only choose the option that has resulted the highest reward in the past. In this case, the convergence is not towards the true optimal strategies but rather towards strategies that are suboptimal yet have been reinforced by the limited exploration strategy.

Based on the illustration Figure 2., we can clearly see the divergence of optimal strategies across agents. Although we cannot derive conclusions from Figure 2., it clearly illustrates the pattern of how different agents have different optimal choices.

In our analysis, Principal Component Analysis (PCA) serves as a method to reduce the dimensionality of our dataset, projecting it onto a space defined by principal components that maximize variance. However, it is important to acknowledge that PCA primarily facilitates visualization and interpretation of the underlying patterns in the data. The projection of the data into a lower-dimensional space, while useful for identifying broad patterns and trends, does not inherently provide detailed insights into the specific characteristics of individual agents within the simulation.

Further, while K -means clustering has been applied to the principal components to identify distinct groups or strategies among agents, it is crucial to recognize the limitations of this approach. K -means clustering categorizes agents into clusters based on their proximity in the PCA-reduced space, which may suggest general strategies or behaviors shared by agents within each cluster. Nonetheless, this method does not directly illuminate the unique attributes or decision-making processes of agents. The clusters identified through K -means are based on the reduced representation of the data, and as such, they offer a simplified overview rather than a comprehensive understanding of agent characteristics.

Therefore, while these analytical techniques—PCA and K -means clustering—provide valuable insights into the overall structure and grouping of agent strategies, they do not permit a granular analysis of agent characteristics beyond the dimensions considered in the PCA. To thoroughly understand the nuances of agent behaviors and strategies, additional analyzes or model interpretations would be necessary, potentially incorporating more detailed agent-specific data or alternative analytical approaches. The PCA with three clusters is illustrated in Figure 4.



Note that in this plot the clusters are numbered from 0 to 2, adhering to Python's zero-based indexing convention. This numbering does not imply that cluster 0 serves as a reference point or holds any particular significance beyond its sequence in the array. Additionally, the K-means clustering algorithm includes elements of randomness due to the initial selection of centroids. At the start of the algorithm, centroids are randomly chosen from the dataset, which serves as the initial group centers. This randomness means that the starting points can significantly influence the final clusters, thereby making this figure just one of many possible outcomes.

5 Discussion

In this study, I explored partner selection dynamics through a highly abstract simulation, acknowledging the limitations in deriving concrete outcomes from such a simplified model. Consistent with intuitive understanding, the simulation’s results suggest that individuals perceived as less attractive experience longer waiting periods before securing a partner. This outcome underscores the significant impact of physical attractiveness on dating success within the model’s framework. I am not able to identify any personality trait that serves as a good predictor of dating success, as personality works as a ‘black-box’ in this model. A notable finding from the simulation is the distinct advantage held by males with the highest attractiveness scores. This group demonstrated the shortest waiting times before engaging in partnerships, indicating the considerable benefit of high physical attractiveness in the simulated dating market. This observation aligns with prevalent societal perceptions and also supports existing research on the significant advantages conferred by physical attractiveness in real-world dating scenarios. This effect is further magnified by the dynamics of competition within the dating pool; attractive males, being a relatively rarer group compared to their female counterparts, face less competition. This scarcity enhances their positioning within the simulated dating market, thereby granting them a distinct advantage in securing partners with minimal waiting time. The findings from my simulations indicate that the aggregate welfare generated by my model achieves on average 71.1% of the welfare realized by the Gale-Shapley algorithm. This suggests that while my approach yields lower overall welfare, it still captures a significant proportion of the potential utility in the simulated dating market. When considering only those individuals who successfully found partners in my model, the algorithm’s performance improves to 81.6% of the Gale-Shapley welfare.

My model has a limitation regarding how agents form their preferences. In this study, I have assumed that agents would settle for their best choice based on their individual experiences from interactions with other agents. This assumption might hold for many people in real life, as people are simply not capable of evaluating potential relationships with people they have not met. However, in reality, many individuals approach relationships with preconceived notions of an ideal partner or relationship dynamics, which influence their dating strategies and partner selection. To encapsulate this aspect of human behavior, future iterations of the model could incorporate a baseline reward parameter. This

parameter would represent the intrinsic value agents assigns to the concept of an ideal relationship, independent of their experiential learning. The extended model would account for idealistic standards or aspirations that individuals may hold independently of their interaction history. In reality, many individuals approach relationships with preconceived notions of an ideal partner or relationship dynamics, which influence their dating strategies and partner selection. These idealistic preferences could, in principle, act as a baseline reward mechanism within the agent’s decision-making process.

My model could be expanded with a complex social network. A social network would serve multiple purposes, such as facilitating interactions and monitoring behaviors. Agents could meet through acquaintances. Additionally, the network could function to punish bad behavior. If agents had options to lie about themselves to gain some advantage (through the implementation of short-term relationships), an infamy system could punish those who deceive others. I have implemented a simple friendship network based on the assumption that inter-gender friendships are more likely to occur. I created a weighted graph, with weights representing both shallow and profound (‘close’) relationships. I have omitted this network in my final simulation, as it seemed redundant for the purpose of this study. It would be interesting to create a dynamic graph that evolves over time, representing how friendships form in real-time. The Barabási-Albert model⁴ (Barabási et al., 1999) with sociometric configurations could be tested for the friendship network.

Further research could implement relationship dynamics. After a decision to date has been made, the relationship between the two agents could be modeled. For example, using stochastic differential equations, ‘Ups and Downs’ in a relationship can be modeled as fluctuations, mathematically represented by, for instance, a Wiener process. Thus, the model can be represented as a system of Stochastic Differential Equations. The core idea would involve agents adjusting their behavior and satisfaction levels in response to the affinity they share with their partners. Such a model necessitates a framework where the utility derived from being in a relationship is superadditive; essentially, each participant

⁴The Barabási-Albert model is a fundamental algorithm for generating scale-free networks, characterized by a power-law degree distribution. Introduced by Albert-László Barabási and Réka Albert in 1999, this model is based on two key principles: growth and preferential attachment. In the BA model, a network starts with an initial connected network of m_0 nodes. New nodes are then added to the network one at a time, each with m edges that link to existing nodes with a probability proportional to the number of links that the existing nodes already have. This ‘rich-get-richer’ mechanism leads to the emergence of hubs — highly connected nodes — which is a common feature in many real-world networks, including social networks.

must find themselves in a better state within the relationship than they would be on their own.

Incorporating societal deviance, such as psychopathy, would be an interesting addition for exploring the impact of antisocial behavior on community dynamics. By simulating the presence of individuals with psychopathic traits, it becomes possible to assess the broader implications of antisocial behavior on the collective well-being and stability of the simulated community. Specifically, modeling the actions of psychopathic individuals allows for an examination of how such behaviors disrupt social cohesion, trust, and relationship formation. Furthermore, this approach facilitates the investigation of a critical threshold: the rate at which the presence of antisocial individuals becomes unsustainable for the community. Beyond a certain point, the detrimental effects of these antisocial agents may outweigh the community’s capacity to adapt or compensate, leading to a breakdown in social order. By quantitatively assessing these dynamics, the model could offer valuable insights into the delicate balance between individual behaviors and community resilience, providing a novel perspective on the consequences of deviance in social contexts.

In future research, the assumption of lifelong partnerships could be reevaluated by allowing agents the flexibility to switch partners post-settlement. This modification could be conceptualized similarly to a swing option in financial modeling, where the option’s holder may choose multiple exercise rights of varying quantities at different times. The adaptation of this model could be implemented as either a discrete or continuous time multiple stopping problem. Bender et al. (2015) offers dual representations for such problems, which vastly reduces complexity and runtime. This adjustment could potentially enhance the realism and applicability of the model by incorporating the dynamics of real-world relationship changes.

In conclusion, my model offers a sophisticated approach to modeling dating scenarios and provides a framework to test network effects⁵ in two-sided markets. Apart from my other findings, I observed that the outcomes of optimal strategies in the dating market depend heavily on types of agents that are present in each group. In my simulation, highly attractive males are scarce, which drives up competition for them among females and creates a negative same-side effect for females. This effect is also observed when there is an imbalance between the numbers of agents on each side of the market, a phenomenon

⁵The network effects discussed here refer to those arising from changes in the agent population, not to be confused with network effects related to the locations within my simulation.

observed in many dating applications where a surplus of male users intensifies competition for potential matches. It is challenging to structure a dating market where Pareto improvement is possible: in most cases, the presence of desirable potential mates on one side of the market creates a positive cross-side effect for individuals on the other side but also increases competition among agents on the former side. Thus, it is often impossible to benefit one side without disadvantaging the other. Dating platforms have to optimize their algorithms to keep up high traffic on their websites. The decision-making process must consider several factors, including which side of the market to benefit, especially under asymmetric network effects. Corporations should identify which groups are most likely to cancel their subscriptions due to insufficient interactions on the website and prioritize determining the optimal gender ratios for each platform. With certain modifications, my simulation could model changes in behavior, incentives, aggregate welfare, and efficiency. An agent-based analysis is especially valuable when altering the matching algorithm of a dating site, thereby predicting the success of such algorithms.

References

- Barabási AL, Albert R, Jeong H (1999) Mean-field theory for scale-free random networks. *Physica A: Statistical Mechanics and its Applications* 272(1):173–187
- Becker S, Cheridito P, Jentzen A (2019) Deep optimal stopping. *Journal of Machine Learning Research* 20(74):1–25
- Bender C, Schoenmakers J, Zhang J (2015) Dual representations for general multiple stopping problems. *Mathematical Finance* 25(2):339–370
- Camargo CQ (2020) New methods for the steady-state analysis of complex agent-based models. *Frontiers in Physics* 8
- Centola D (2015) The social origins of networks and diffusion. *American Journal of Sociology* 120(5):1295–1338
- De La Mare JK, Lee AJ (2023) Assortative preferences for personality and online dating apps: Individuals prefer profiles similar to themselves on agreeableness, openness, and extraversion. *Personality and Individual Differences* 208:112,185
- Digman JM (1990) Personality structure: Emergence of the five-factor model. *Annual review of psychology* 41(1):417–440
- Eriksson K, Sjöstrand J, Strimling P (2007) Optimal expected rank in a two-sided secretary problem. *Operations Research* 55(5):921–931
- Ery J, Michel L (2021) Solving optimal stopping problems with deep q-learning. *arXiv:2101.09682*
- Gale D, Shapley LS (1962) College admissions and the stability of marriage. *The American Mathematical Monthly* 69(1):9–15
- Goodfellow I, Bengio Y, Courville A (2016) *Deep Learning*. MIT Press
- Ha T, Overbeek G, Engels R (2010) Effects of attractiveness and social status on dating desire in heterosexual adolescents: An experimental study. *Archives of Sexual Behavior* 39(5):1063–1071

- Ibáñez-Berganza M, Amico A, Loreto V (2019) Subjectivity and complexity of facial attractiveness. *Scientific Reports* 9(1):8364
- Jacka SD (1991) Optimal stopping and the american put. *Mathematical Finance* 1(2):1–14
- Ligmann-Zielinska A, Siebers PO, Magliocca NR, Parker DC, Grimm V, Du J, Cenek M, Radchuk V, Arbab NN, Li S, Berger U, Paudel R, Robinson DT, Jankowski P, An L, Ye X (2020) One size does not fit all: A roadmap of purpose-driven mixed-method pathways for sensitivity analysis of agent-based models. *Journal of Artificial Societies and Social Simulation* 23(1):6
- Neyt B, Baert S, Vandenbulcke S (2020) Never mind i’ll find someone like me – assortative mating preferences on tinder. *Personality and Individual Differences* 155:109,739
- Shaheen JAE, Henley C, McKenna L, Hoang S, Abdulwahab F (2022) Comparative agent-based simulations on levels of multiplicity using a network regression: A mobile dating use-case. *Applied Sciences* 12(4)
- Sutton RS, Barto AG (2018) Reinforcement learning: An introduction. MIT press
- Voigt S, Hinz O (2015) Network effects in two-sided markets: why a 50/50 user split is not necessarily revenue optimal. *Business Research* 8(1):139–170

Appendix

5.1 Pseudocodes

Algorithm 1 Agent Class

```
1: Class Agent
2: Properties:
3: id, action_space, q_table, alpha, gamma, epsilon, gender, name
4: appearances, beauty_overall, innerqs, appearance_weight, innerq_weight
5: interaction, preferences, type, met_agents, utility_history, learning_rate
6: selected_partner, location, location_history, visited_locations, active, settle_score,
   settle_decay
7: End Properties
8:
9: function CONSTRUCTOR(id, gender, seed, action_space, state_space, alpha, gamma, epsilon)
10:   Initialize properties
11:   this.q_table  $\leftarrow$  np.full(state_space, len(action_space), 0.1)
12:   POINTS_ROLL(seed)
13: end function
14:
15: function RESET_AGENT_STATE
16:   Reset agent's state for a new simulation run
17: end function
18:
19: function __REPR__
20:   Provide a string representation of the agent
21:   return agent's ID, gender, and beauty overall
22: end function
23:
24: function POINTS_ROLL(seed)
25:   Roll points for appearances and inner qualities based on seed
26: end function
27:
28: function WEIGHTS_INITIAL(seed)
29:   Initialize weights for appearances and inner qualities
30: end function
31:
32: function GENERATE_PREFERENCES(others)
33:   Generate preference list based on utilities calculated for other agents
34: end function
35: End Class
```

Algorithm 2 Location Decision

```
1:  $locations \leftarrow \{(85, 70), (50, 90), (15, 70), (15, 30), (50, 10), (85, 30), (50, 50)\}$ 
2: procedure UPDATEHISTORY( $self, location, utility$ )
3:   Append ( $location, utility$ ) to  $self.location\_history$ 
4: end procedure
5: procedure DECIDENEXTLOCATION( $self, locations$ )
6:    $best\_score \leftarrow -\infty$ 
7:    $best\_location \leftarrow None$ 
8:   for  $loc$  in  $locations$  do
9:      $historical\_utility\_score \leftarrow$  sum of utilities for  $loc$  in  $self.location\_history$ 
10:    if  $historical\_utility\_score > best\_score$  then
11:       $best\_score \leftarrow historical\_utility\_score$ 
12:       $best\_location \leftarrow loc$ 
13:    end if
14:  end for
15:  return  $best\_location$ 
16: end procedure
```

Algorithm 3 Utility Calculation

```
1: procedure CALCULATEUTILITY( $self, other\_appearance, other\_innerq$ )
2:    $appearance\_utility \leftarrow self.appearance\_weight @ other\_appearance$ 
3:    $innerq\_utility \leftarrow -np.mean(np.square(self.innerqs - other\_innerq))$ 
4:    $utility \leftarrow appearance\_utility + innerq\_utility$ 
5:   return  $utility$ 
6: end procedure
   where @ refers to matrix multiplication
```

Algorithm 4 Softmax and Boltzmann Action Selection

```
1: procedure SOFTMAX( $q\_values, \tau$ )
2:    $preferences\_ \leftarrow q\_values / \tau$ 
3:    $max\_preference \leftarrow \max(preferences\_)$ 
4:    $exp\_preferences \leftarrow \exp(preferences\_ - max\_preference)$ 
5:    $sum\_exp\_preferences \leftarrow \sum(exp\_preferences)$ 
6:    $action\_probs \leftarrow exp\_preferences / sum\_exp\_preferences$ 
7:   return  $action\_probs$ 
8: end procedure
9:
10: procedure CHOOSEACTIONBOLTZMANN( $self, state, \tau$ )
11:    $q\_values \leftarrow self.q\_table[state]$ 
12:    $action\_probs \leftarrow SOFTMAX(q\_values, \tau)$ 
13:    $action\_index \leftarrow$  random choice from  $action\_probs$ 
14:    $self.current\_action\_index \leftarrow action\_index$ 
15:    $self.settle\_decay \leftarrow self.action\_space[action\_index]$ 
16:   return  $self.settle\_decay, action\_index$ 
17: end procedure
```

Algorithm 5 Simulated Interaction and Decision Making Process: Part 1

```
1: procedure SIMDECISIONSPARTONE(males, females, locations)
2:   np.random.seed(52)
3:   for all agent  $\in$  males + females do
4:     agent.initialize_location()
5:     agent.interaction_lim  $\leftarrow$  agent.set_interaction_limits()
6:   end for
7:   decisions  $\leftarrow$  []
8:   to_deactivate  $\leftarrow$  set()
9:   for all location  $\in$  locations do
10:    active_males  $\leftarrow$  [male for male  $\in$  males if male.location =
      location and male.active]
11:    active_females  $\leftarrow$  [female for female  $\in$  females if female.location =
      location and female.active]
12:    if not active_females or not active_males then
13:      continue
14:    end if
15:    initial_list  $\leftarrow$  np.random.permutation([female.id for female  $\in$ 
      active_females]).tolist()
16:    male_assignments  $\leftarrow$  {}
17:    for all male  $\in$  active_males do
18:      rotation_index  $\leftarrow$  male.id mod len(initial_list)
19:      rotated_list  $\leftarrow$  initial_list[-rotation_index :] + initial_list[:
      -rotation_index]
20:      male_assignments[male.id]  $\leftarrow$  rotated_list[: male.interaction_lim]
21:    end for
22:    female_lists  $\leftarrow$  {female.id : [] for female  $\in$  active_females}
23:    for male_id, assigned_female_ids in male_assignments.items() do
24:      for all female_id in assigned_female_ids do
25:        female  $\leftarrow$  next((f for f  $\in$  active_females if f.id =
      female_id), None)
26:        if female and len(female_lists[female_id]) <
      female.interaction_lim then
27:          female_lists[female_id].append(male_id)
28:        end if
29:      end for
30:    end for
```

Algorithm 6 Simulated Interaction and Decision Making Process: Part 2

```
31:     for male_id, female_ids in male_assignments.items() do
32:         male  $\leftarrow$  next(m for m  $\in$  active_males if m.id = male_id), None)
33:         if male and male.active then
34:             for all female_id in female_ids do
35:                 female  $\leftarrow$  next(f for f  $\in$  active_females if f.id =
female_id), None)
36:                 if female and female.active then
37:                     m_utility  $\leftarrow$  male.interact(female)
38:                     f_utility  $\leftarrow$  female.interact(male)
39:                     if male.decide_to_stop() and female.decide_to_stop() then
40:                         decisions.append((male.id, female.id))
41:                         to_deactivate.update({male.id, female.id})
42:                         break
43:                     end if
44:                 end if
45:             end for
46:         end if
47:     end for
48:     for all agent in males + females do
49:         if agent.id in to_deactivate then
50:             agent.active  $\leftarrow$  False
51:         end if
52:     end for
53:
54:     return decisions
```

Algorithm 7 Update Location History and Decide Next Location

```
1: procedure UPDATEHISTORY(self, location, utility)
2:     self.location_history.append((location, utility))
3: end procedure
4:
5: procedure DECIDENEXTLOCATION(self, locations)
6:     best_score  $\leftarrow -\infty$ 
7:     best_location  $\leftarrow$  None
8:     for loc in locations do
9:         historical_utility_score  $\leftarrow$  sum(utility for (location, utility) in
self.location_history if location == loc)
10:        if historical_utility_score > best_score then
11:            best_score  $\leftarrow$  historical_utility_score
12:            best_location  $\leftarrow$  loc
13:        end if
14:    end for
15:    return best_location
16: end procedure
```

Algorithm 8 Gale-Shapley Algorithm for Stable Marriage: Part 1

```
1: function GALESHAPLEYPARTONE(prefer)
2:    $N \leftarrow 500$ 
3:    $wPartner[N] \leftarrow$  array filled with  $-1$ 
4:    $mFree[N] \leftarrow$  array filled with False
5:    $freeCount \leftarrow N$ 
6:    $pairs \leftarrow []$ 
7:   while  $freeCount > 0$  do
8:      $m \leftarrow 0$ 
9:     while  $m < N$  and  $mFree[m] == False$  do
10:       $m \leftarrow m + 1$ 
11:   end while
12:    $i \leftarrow 0$ 
13:   while  $i < N$  and  $mFree[m] == False$  do
14:      $w \leftarrow prefer[m][i]$ 
15:     if  $wPartner[w - N] == -1$  then
16:        $wPartner[w - N] \leftarrow m$ 
17:        $mFree[m] \leftarrow True$ 
18:        $freeCount \leftarrow freeCount - 1$ 
19:        $pairs.append((m, w))$ 
20:     else
21:        $m1 \leftarrow wPartner[w - N]$ 
22:       if  $WPREFERSM1OVERM(prefer, w, m, m1) == False$  then
23:          $wPartner[w - N] \leftarrow m$ 
24:          $mFree[m] \leftarrow True$ 
25:          $mFree[m1] \leftarrow False$ 
26:          $pairs \leftarrow REPLACEPAIR(pairs, m1, m, w)$ 
27:       end if
28:     end if
29:      $i \leftarrow i + 1$ 
30:   end while
31: end while
```

Algorithm 9 Gale-Shapley Algorithm for Stable Marriage: Part 2

```
32:  function WPREFERSM1OVERM(prefer, w, m, m1)
33:    for i from 0 to  $N - 1$  do
34:      if prefer[w][i] == m1 then
35:        return True
36:      else if prefer[w][i] == m then
37:        return False
38:      end if
39:    end for
40:  end function
41:
42:  function REPLACEPAIR(pairs, m1, m, w)
43:    for each pair in pairs do
44:      if pair[0] == m1 then
45:        pair  $\leftarrow$  (m, w)
46:      end if
47:    end for
48:    return pairs
49:  end function
```

Algorithm 10 Find Partner, Compare Preferences, and Compare Outcomes

```
1: procedure GETPARTNERFORAGENT(agent_id, pairs)
2:   for each pair in pairs do
3:     if pair[0] == agent_id then
4:       return pair[1]
5:     else if pair[1] == agent_id then
6:       return pair[0]
7:     end if
8:   end for
9:   return None
10: end procedure
11:
12: procedure COMPAREPREFERENCE RANK(agent, partner_id)
13:   if partner_id is None then
14:     return  $\infty$ 
15:   else
16:     return agent.preferences.index(partner_id)
17:   end if
18: end procedure
19:
20: procedure COMPAREOUTCOMES(stopping_decisions, gale_shapley_pairs, agents)
21:   better_than_gs  $\leftarrow$  []
22:   worse_than_gs  $\leftarrow$  []
23:   same_as_gs  $\leftarrow$  []
24:   no_partner  $\leftarrow$  []
25:   for each agent in agents do
26:     partner_id  $\leftarrow$  GETPARTNERFORAGENT(agent.id, stopping_decisions)
27:     gs_partner  $\leftarrow$  GETPARTNERFORAGENT(agent.id, gale_shapley_pairs)
28:     stopping_rank  $\leftarrow$  COMPAREPREFERENCE RANK(agent, partner_id)
29:     gs_rank  $\leftarrow$  COMPAREPREFERENCE RANK(agent, gs_partner)
30:     if partner_id is None then
31:       no_partner.append(agent.id)
32:       continue
33:     end if
34:     if stopping_rank < gs_rank then
35:       better_than_gs.append(agent.id)
36:     else if gs_rank < stopping_rank then
37:       worse_than_gs.append(agent.id)
38:     else
39:       same_as_gs.append(agent.id)
40:     end if
41:   end for
42:   return better_than_gs, worse_than_gs, same_as_gs, no_partner
43: end procedure = 0
```

Algorithm 11 Simulation Process

```
1: Class Simulation
2: Properties:
3: duration, current_time, decisions, num_states, agents
4: End Properties
5:
6: function CONSTRUCTOR(duration, num_states)
7:   this.duration  $\leftarrow$  duration
8:   this.num_states  $\leftarrow$  num_states
9:   this.current_time  $\leftarrow$  0
10:  this.decisions  $\leftarrow$  []
11: end function
12:
13: function INITIALIZEAGENTS
14:   Initialize males and females agents
15:   this.agents  $\leftarrow$  males  $\cup$  females
16:   For each agent in agents, initialize their location and interaction limits
17:   return males, females
18: end function
19:
20: function CURRENTSTATE
21:   state_duration  $\leftarrow$  this.duration  $\div$  this.num_states
22:   return min(this.current_time  $\div$  state_duration, this.num_states - 1)
23: end function
24:
25: function RUNSIMULATIONSTEP
26:   current_state  $\leftarrow$  CURRENTSTATE
27:   For each active male and female in their respective locations, make decisions
28:   Update this.decisions with the new decisions
29:   for each agent in agents do
30:     Determine if agent finds a partner and interact accordingly
31:     Update Q-table based on interaction results
32:   end for
33: end function
34:
35: function UPDATE
36:   RUNSIMULATIONSTEP
37:   Increment this.current_time
38: end function
```

Algorithm 12 Simulation Process Second Part

```
1:
2: function RUN
3:   Initialize temperature parameters for Boltzmann exploration
4:   for  $tick \leftarrow 1$  to  $this.duration$  do
5:      $current\_state \leftarrow$  CURRENTSTATE
6:     For each agent, choose an action based on the Boltzmann exploration
7:     UPDATE
8:     Adjust temperature parameter
9:   end for
10:  Print decisions made during the simulation
11: end function
12:
13: function PLOTHISTOGRAM
14:  Plot histogram of beauty overall scores
15: end function
16:
17: function PLOTUTILITY
18:  Plot utility over time for a specific agent
19: end function
20:
21: function RESETSTATE
22:  Reset simulation to initial state
23: end function
24:
25: function LOCATIONINNERQS
26:  Calculate and return inner quality scores for each location
27: end function
28:
29: function RUNANDANALYZE
30:  Analyze and print final location qualities
31: end function
32: End Class
```

Algorithm 13 Run Simulation and Export Q-tables to Excel

```
1: function RUNSIMULATION
2:   Initialize simulation with males and females agents
3:   for  $run \leftarrow 1$  to  $n\_runs$  do
4:     Reset simulation and agent states
5:     if  $run = n\_runs$  then
6:       Generate preferences for males and females
7:     end if
8:     Run simulation steps
9:     Update location inner qualities
10:  end for
11:  Generate Gale-Shapley pairs
12:  Compare outcomes to determine better agents, worse agents, and same agents
13:  Print results
14:  Export Q-tables to Excel
15:  Plot histogram and utility graph
16: end function
17:
18: function EXPORTQTABLESTOEXCEL(fileName)
19:   Prepare data from agents' Q-tables, interaction limits, and other attributes
20:   Create DataFrame from data
21:   Set DataFrame columns for Q-Values, interaction limits, genders, and other at-
   tributes
22:   Export DataFrame to Excel file
23:   Print confirmation message
24: end function
```

Algorithm 14 PCA on Dating Strategies

```
1: Load data from 'aqn2.xlsx' into DataFrame  $df$ 
2: Define features as {'Interaction Limit', 'Inner Qualities 1', ..., 'Inner Quality Weights 3'}
3: Standardize features from  $df$  into  $X$ 
4: Initialize PCA with  $n\_components = 3$ 
5: Fit  $X$  to PCA and transform into principalComponents
6: Create principalDf from principalComponents with columns {'Principal Component 1', 'Principal Component 2', 'Principal Component 3'}
7: Print explained variance ratio
8: for each component in first three components do
9:   Print loadings for component
10: end for
11: Concatenate principalDf with agent_id from  $df$  into finalDf
12: Save finalDf to 'pca_output.xlsx'
13: Plot 3D scatter of Principal Components
```
