UDC 004.9
DOI: https://doi.org/10.36994/2788-5518-2024-01-07-15

RULE BASED MATCHMAKING SYSTEM

Sergii Suglobov, *Open International University of Human Development «Ukraine», Kyiv, Ukraine. https://orcid.org/0009-0004-6784-0968, s.suglobov@gmail.com*

Anatolii Tymoshenko*, Ph.D., Ass Prof., Open International University of Human Development «Ukraine», Kyiv, Ukraine. https://orcid.org/0000-0003-0954-3186, timoshag@i.ua*

Abstract. *In modern online gaming, the issue of effective matchmaking is becoming increasingly important for ensuring fairness and player satisfaction. As the number of participants in a game grows, it is necessary to develop systems that can not only quickly and accurately determine team compositions but also consider numerous factors that impact the quality of the gameplay. This article presents an approach to building a rule-based matchmaking system. It combines algorithmic logic and expert knowledge to allocate players into teams based on predefined criteria, such as skill level, geographic location, preferences in game modes, and other parameters.*

The article discusses the operation of the system, which includes creating and processing complex metadata structures known as query schemas. In the initial stage, the system collects all player requests, sorts them by priority and creation time, and then applies various rules to calculate similarity matrices between these requests. Each rule computes the similarity between pairs of elements from the source and target schema, creating a similarity matrix where each element reflects the degree of similarity between two elements. The results of these matrices are then aggregated to create a unified similarity matrix, from which the system selects the most similar requests to form teams. The final stage involves evaluating the proposed matchmaking options to ensure the quality and balance of the gameplay.

The system is also adaptable to various matchmaking scenarios due to the use of rules that can dynamically alter the system's behavior depending on the characteristics of the input data. This improves the accuracy and efficiency of the matchmaking process. As a result, the developed matchmaking system is a versatile tool for creating balanced teams that meet the demands of different gaming environments.

Key words: *matching system, matching rules, schema matching.*

СИСТЕМА МАТЧМЕЙКІНГУ НА ОСНОВІ ПРАВИЛ

Суглобов С. В.*, Відкритий міжнародний університет розвитку людини «Україна», Київ, Україна. https://orcid.org/0009-0004-6784-0968, s.suglobov@gmail.com*

Тимошенко А. Г.*, к.т.н., доц., Відкритий міжнародний університет розвитку людини «Україна», Київ, Україна. https://orcid.org/0000-0003-0954-3186, timoshag@i.ua*

Анотація. *У сучасному онлайн-геймінгу питання ефективного матчмейкінгу стає усе більш важливим для забезпечення справедливості та задоволення гравців. В умовах, коли кількість учасників у грі зростає, необхідно створювати системи, здатні не лише швидко та точно визначати склади команд, а й ураховувати безліч чинників, які впливають на якість гри. У статті представлено підхід до побудови системи матчмейкінгу, що базується на правилах. Вона поєднує у собі алгоритмічну логіку та знання експертів, забезпечуючи розподіл гравців у команді на основі попередньо заданих критеріїв, таких як рівень навичок, географічне розташування, переваги в ігрових режимах та інші параметри.*

Розглянуто процес роботи системи, який включає створення та обробку складних структур метаданих, відомих як схеми запитів. На першому етапі система отримує усі запити гравців, сортує їх за пріоритетом та часом створення, після чого застосовує різні правила для обчислення матриць схожості між цими запитами. Кожне правило обчислює схожість між парами елементів із вихідної та цільової схем, створюючи матрицю схожості, де кожен елемент матриці відображає ступінь подібності між двома елементами. Далі результати цих матриць агрегуються для створення єдиної підсумкової матриці схожості, з якої система відбирає найбільш схожі запити для формування команд. Останнім етапом є оцінка запропонованих варіантів матчування для забезпечення якості та збалансованості гри.

Система також здатна адаптуватися до різних сценаріїв матчмейкінгу завдяки використанню правил, які можуть динамічно змінювати поведінку системи залежно від характеристик вхідних даних. Це дає змогу поліпшити точність та ефективність процесу матчування. Таким чином, розроблена система матчмейкінгу є універсальним інструментом для створення збалансованих команд, що відповідають вимогам різних геймерських середовищ.

Ключові слова: *система матчмейкінгу, правила матчмейкінгу, відповідність схеми.*

Introduction

Player request represents a comprehensive set of data encompassing a player's preferences, constraints, and requirements for online gaming or just a complex data structure for system to operate. It includes information such as game and mode preferences, skill levels, geographical considerations, and other relevant factors. The matchmaking system utilizes this data to create balanced and enjoyable team compositions tailored to individual players' needs.

Matching complex metadata structures is crucial in various domains often referred to as schema matching or ontology alignment. To expedite this task, schema matching systems have to be developed. These systems leverage matching algorithms, also known as matchers, to analyze the structural and semantic similarities between schema elements or instance data. However, the effectiveness of these systems heavily relies on the incorporation of domain-specific knowledge and rules.

Rules play a crucial role in schema matching systems by guiding the matching process and shaping the decisions made by the algorithm. These rules encapsulate expert knowledge and heuristics, enabling the system to interpret and adapt to diverse matching scenarios. For instance, a rule might prioritize structural similarities over semantic ones in certain contexts, or it might adjust matching thresholds based on the characteristics of the input schemes.

By integrating rules into the matching process, schema matching systems can enhance their accuracy, efficiency, and adaptability. Rules provide a means to encode domain-specific insights and preferences, empowering the system to navigate complex matching challenges effectively. Moreover, rules enable the system to dynamically adjust its behavior based on the characteristics of the input data, leading to more robust and reliable matching outcomes.

In essence, the incorporation of rules into schema matching systems represents a critical step towards developing efficient and adaptable solutions capable of addressing the diverse matching needs across different domains. These rules serve as the guiding principles that govern the system's decision-making process, ensuring that it can effectively handle various matching scenarios and produce high-quality mapping suggestions.

Workflow

A matchmaking request or schema comprises various schema elements, each characterized by a name, value with different data type. The scope of schemes is broad and encompasses diverse metadata structures. The objective of a schema matching system is to derive mapping suggestions between a given source schema (S) and a target schema (T).

To accomplish this, most schema matching systems employ a combination of matching operators, including rules, aggregation, and selection. Rules compute similarity values for pairs of schema elements from S and T, generating a similarity matrix of size |S|*|T|. Each entry in this matrix represents the similarity between two elements, ranging from 0 (low similarity) to 1 (high similarity).

So, each rule computes a similarity value for all elements from the source and the target request and constructs a similarity matrix of size |S|*|T| as output.

Matching sequence:

1. Get all requests, sorted by priority and creation time.
2. Compute similarity matrix of all requests by each rule that responsible by specified values.
3. Combines results of rules similarity matrix to a single aggregated similarity matrix.
4. Iterate over aggregated matrix and select the most similar requests.
5. Evaluate final match proposal.

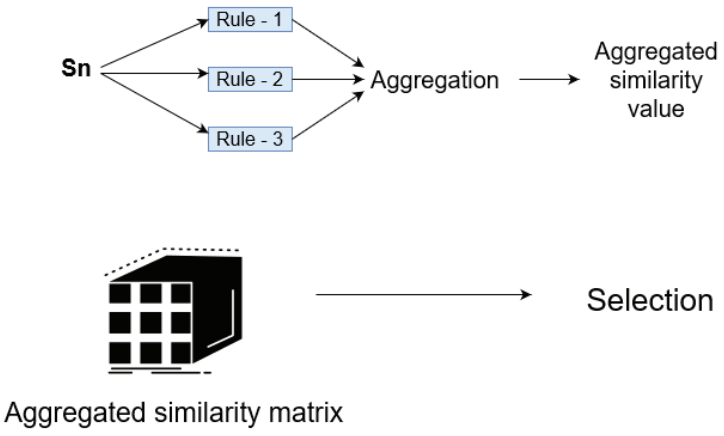


Fig. 1. Processing request schemes using matrices

To enhance the quality of matching results, aggregation operators are utilized to consolidate the output of multiple rules into a single aggregated similarity matrix. Subsequently, selection operators are employed to identify the most probable element pairs, typically using strategies such as Threshold, MaxN, etc.

– **Threshold**: Filters entries above a specified threshold.

– **MaxN**: Returns the N highest entries in each row or column.

Selection strategies accommodate scenarios involving n:m mappings, allowing elements to participate in multiple correspondences. From the selected matrix, a mapping between the source schema S and the target schema T is constructed, comprising a set of correspondences (s, t, sim) denoting source and target elements along with their similarity values.

From aggregated matrix we will try to select the most similar players.

– Exclude all zero values

– For better matches it is recommended to use threshold for final selection.

Selection produce matching proposals that should be evaluated to finalize proposal, because among 3 players with similarity 0.5 they can be totally different, for example we can get a raise condition situation when 3 players can't be intersected by territory, like in example: (eu, us), (us, cis), (cis, eu), we have a situation when for this players we can't chose a territory that suits them.

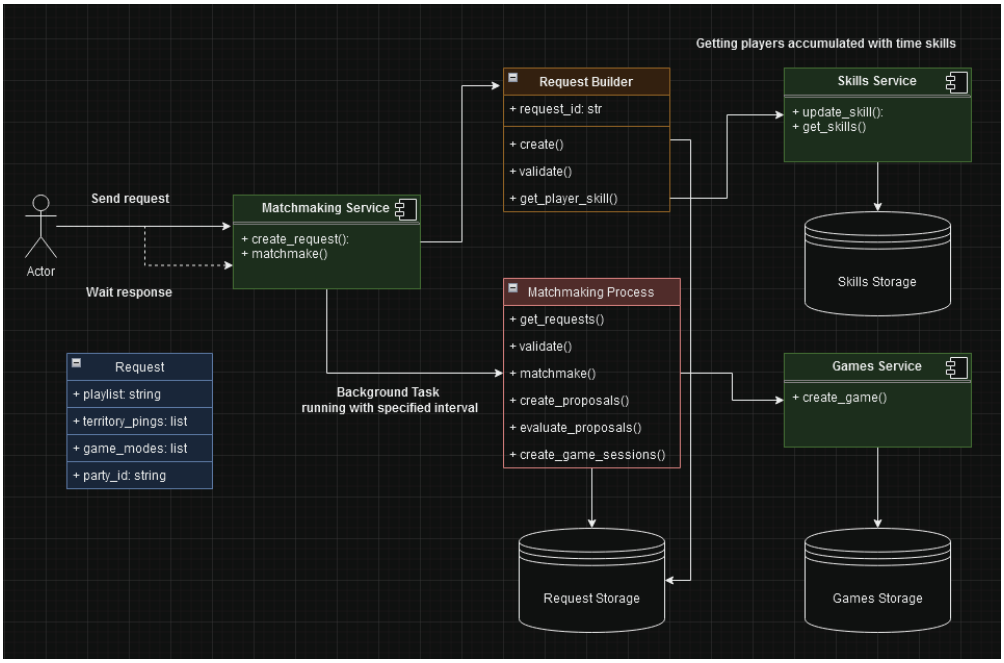


Fig. 2. Components high level vision

Matching Rule

The Rule serves as a fundamental component within the schema matching system, encapsulating specific guidelines or directives governing the matching process. Essentially, a Rule object embodies expert knowledge or heuristics aimed at guiding the system's decision-making and adaptation capabilities.

A matching rule comprises the following components:

– **Pattern**: This component delineates a segment of the process graph where the rule can be applied. The pattern may be empty, particularly within rules initiating the process construction.

– **Action**: The action defines a prescribed alteration to instances of the identified pattern. This encompasses additions or modifications of one or multiple operators within the process.

– **Relevance Function**: Responsible for computing the relevance of the rule within the current matching process and match task. Leveraging computed schema and matrix features from the input schemas and existing similarity matrices, the relevance function generates a relevance value ranging from 0 to 1. This value dictates whether the rule is executed.

– **Optional Check Function**: This function evaluates the quality of the changes introduced by the action subsequent to rule application. It also utilizes matrix features to compute a value between 0 and 1, serving as an assessment of the alterations' efficacy.

To better explain the parts of a matching rule we introduce few simple examples (it is not production ready examples, just to get an idea):

Matching by territories

Assume that we need to match players over the world with many territories and ping (important parameter in games) is different for every player, so from full list of territories player can choose only one/few.

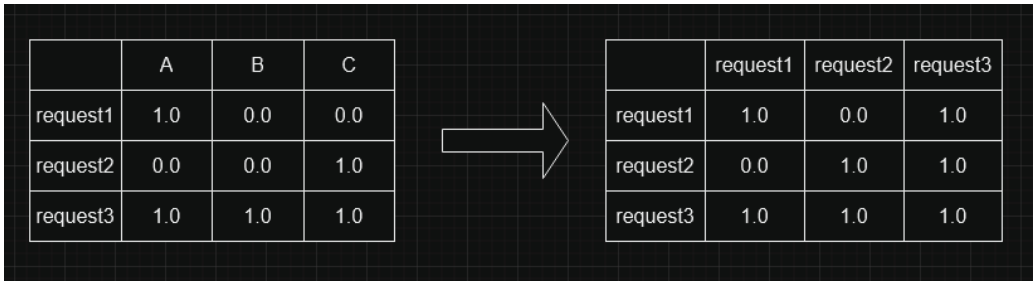
```
import pandas as pd

ALL_TERRITORIES = ['A', 'B', 'C']

def close_territories(requests: list[Request]) -> pd.DataFrame:
    request_ids = [req.id for req in requests]
    df = pd.DataFrame([

        [1 if t in req.territories else 0 for t in ALL_TERRITORIES] for req in requests],

        index=request_ids, columns=ALL_TERRITORIES)
    return df.dot(df.T)
```



	A	B	C
request1	1.0	0.0	0.0
request2	0.0	0.0	1.0
request3	1.0	1.0	1.0

	request1	request2	request3
request1	1.0	0.0	1.0
request2	0.0	1.0	1.0
request3	1.0	1.0	1.0

Fig. 3. Converting raw territory data to mapping among player requests

Here is a brief summary of the function:

– This function computes a similarity matrix based on the proximity of territories between different requests.

– It creates a DataFrame where each row corresponds to a request and each column represents a territory. The cell values are binary, indicating whether a request includes a particular territory.

– The function then calculates the dot product of the DataFrame with its transpose, resulting in a square matrix where each element represents the number of common territories between two requests.

– Normalization is performed to scale the values between 0 and 1, ensuring uniformity across different datasets.

– The resulting similarity matrix is returned.

Matching by skill

Assume that we need to match players that have different skills, all players are unique, some player may play more time, etc.

The idea of every rule is particularly the same, we need to normalize initial data, map it on other players and get players similarity among each other.

```
import pandas as pd

def fair_skill(requests: list[Request]) -> pd.DataFrame:
    skills = [req.skill for req in requests]
    min_s, max_s = min(skills), max(skills)
    request_ids = [req.id for req in requests]
    df = pd.DataFrame([skills] * len(skills), index=request_ids, columns=request_ids)
    df = (df - min_s) / (max_s - min_s)
    df = df.sub(df.T).abs()
    df = 1 / (1 + df)
    return df
```

Here is a brief summary of the function:

– This function computes a similarity matrix based on the fairness of skill levels between different requests.

– It creates a DataFrame where each row and column represent a request, with the cell values indicating the skill level of each request.

– The function normalizes the skill levels within the range [0, 1] based on the minimum and maximum skill levels among all requests.

– It calculates the absolute difference between the normalized skill levels of each pair of requests and computes the reciprocal to obtain similarity values. Higher skill differences result in lower similarity scores.

– The resulting similarity matrix is returned.

Both functions utilize the Pandas library to handle data manipulation efficiently, providing a convenient interface for working with tabular data structures.

Take a look on the example of matching three requests in that way:

Table 1

Explaining rule results

x	Territories			Skills			Aggregated		
x	request1	request2	request3	request1	request2	request3	request1	request2	request3
request1	1.0	0.0	0.5	1.0	0.66	0.83	1.0	0.0	0.64
request2	0.0	1.0	0.5	0.66	1.0	0.58	0.0	1.0	0.54
request3	0.5	1.0	1.0	0.83	0.58	1.0	0.64	0.76	1.0

After applying rules to requests we will get resulted aggregated matrices that show us next:

– Request1 is incompatible with Request2

– Request1 is better matching with Request3 than Request2

– Request3 is compatible with both other.

– Resulted value in 0 mean that players can't play together at all, this parameter can be further optimized by adding threshold value for example equal to 0.2-0.4 to skip all resulted similarity values.

Aggregate rules

Aggregation rules enhance a process by introducing aggregation operators and consolidating dangling operators from a matching process. These dangling operators may have been generated by initiating rules. Various aggregation operators are available, including AVERAGE, OWA, HARMONY, or MIN/MAX, each offering distinct advantages and drawbacks.

In my works I use weighted geometric average to derive a final matrix from multiple matrices. Utilizing a weighted geometric average offers several advantages, including the ability to incorporate multiple sources of information while considering their relative importance. Additionally, it provides a flexible framework for combining diverse matrices, enabling robust aggregation of similarity scores in the context of schema matching or other matching tasks.

Example using pure numpy (as np):

```
np.exp(np.average(np.log(matrices), axis=0, weights=np.array(self._rule_weights)))
```

or scipy library can be used: from scipy.stats.mstats import gmean

Selection

The selection process from an aggregated matrix and splitting on teams involves identifying the most suitable correspondences between elements from the source and target schemas, followed by grouping these correspondences into teams based on specific criteria. Here's a description of the combined selection and team-splitting process:

Aggregated Matrix: Start with the aggregated similarity matrix, which combines the similarity scores from multiple individual matrices. Each row of the matrix is a players sorted by creation time, so we need to iterate over each row and check if we have enough players to create a game for them.

Thresholding: Optionally, apply a threshold to the values in the aggregated matrix to filter out less significant similarity scores. This helps focus on the most relevant correspondences.

Team Formation: Once the correspondences are identified, proceed to split them into teams based on predetermined criteria. This could involve factors such as balancing skill levels, ensuring diversity in roles or positions, or maximizing overall team cohesion.

Team Splitting Criteria: Determine the criteria for splitting correspondences into teams. This could include factors such as:

– Skill Level: Ensure that each team has a balanced distribution of skill levels to promote fair competition.

Matchmaking flow

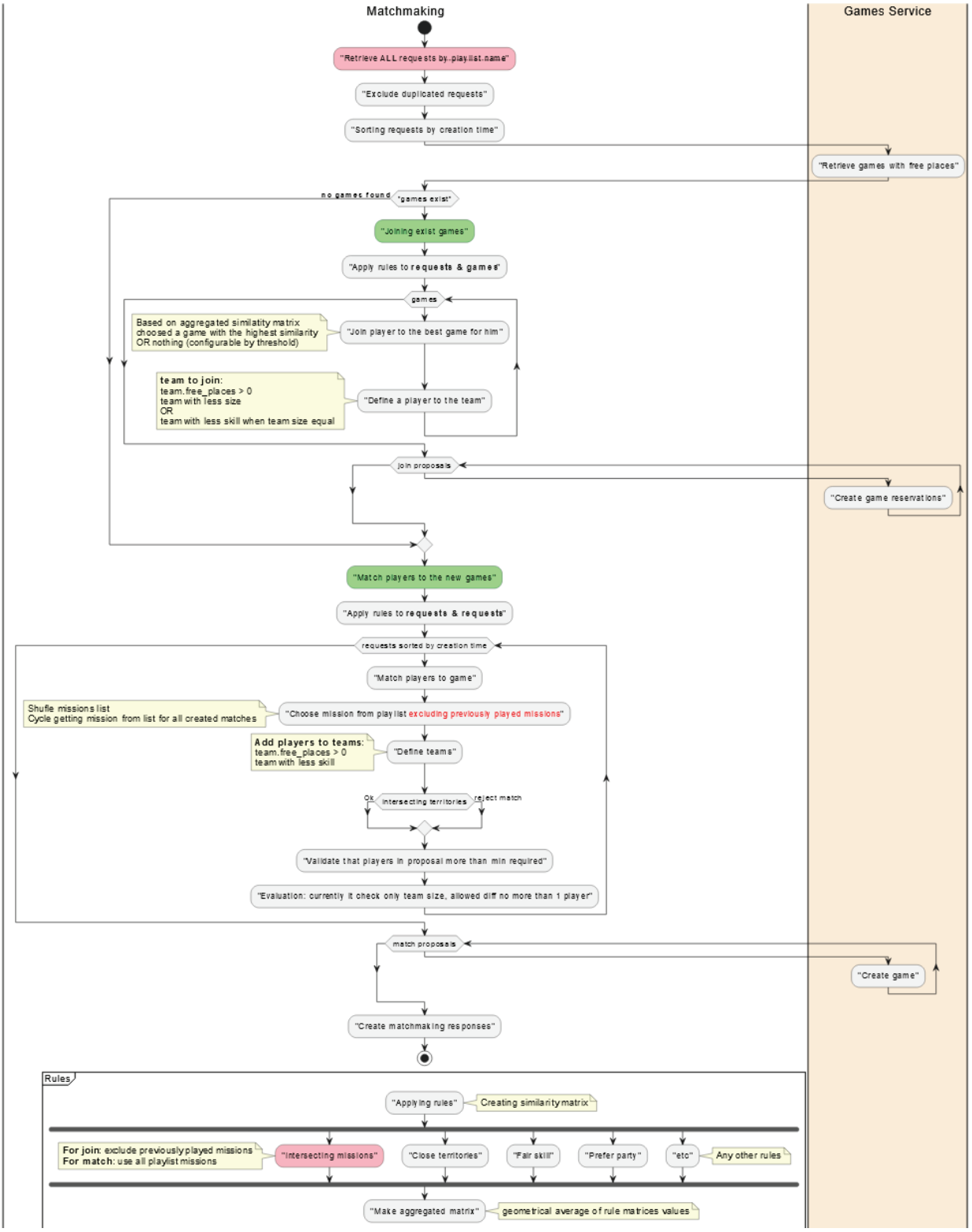


Fig. 4. Full matchmaking flow diagram

– Role Diversity: Ensure that each team has a diverse mix of roles or positions to facilitate strategic gameplay.
– Communication Preferences: Consider players' communication preferences
– Etc

Final Mapping and Team Assignment: Construct a final mapping between the source and target schemas, incorporating the team assignments. This mapping represents the recommended associations between elements from the two schemas, grouped into teams based on the specified criteria.

Evaluation

In evaluating the final mapping result, several key aspects need consideration to ensure the fairness, diversity, and overall quality of the gameplay experience. Here's a breakdown of the evaluation process, encompassing various criteria and potential adjustments:

Re-balancing teams:
– Evaluate the distribution of skill levels, roles, or other relevant factors across the teams.
– If significant imbalances are detected, consider re-balancing teams to ensure fair competition.
– This may involve redistributing players, adjusting team compositions, or implementing dynamic balancing mechanisms during gameplay.

Randomizing:
– Assess the variety and suitability of game modes assigned to each team.
– Introduce randomness or rotation in game mode selection to prevent monotony and enhance player engagement.

Raising conditions for choosing territory:
– Review the criteria for selecting territories and assess their impact on gameplay dynamics.
– Consider raising the conditions for choosing territories to introduce strategic depth and challenge.
– This could involve implementing additional criteria such as territorial control objectives, resource allocation, or strategic advantages/disadvantages.

Continuous monitoring and iterative improvement:
– Continuously monitor gameplay metrics, player feedback, and performance data to identify areas for improvement.
– Iterate on the matchmaking and team allocation algorithms based on real-world observations and feedback from players.
– Strive for a balance between fairness, competitiveness, and enjoyment, and be prepared to adapt the matchmaking system accordingly to meet evolving player needs and expectations.

By carefully evaluating the final mapping result and implementing adjustments as necessary, you can ensure a balanced, diverse, and engaging gameplay experience for all participant

Conclusion

The rule-based matchmaking system presented in this article offers a promising approach to enhancing the multiplayer gaming experience. By intelligently allocating players into teams based on predefined criteria and leveraging matching rules to determine player similarity and construct similarity matrices, the system strives to promote fairness, balance, and enjoyment in gaming environments. Through the aggregation of similarity matrices and the selection of optimal match proposals, the system aims to optimize team compositions and ultimately improve the quality of matchmaking outcomes. Additionally, empirical evaluation and real-world testing are essential to validate the effectiveness and robustness of the system across diverse gaming scenarios and player demographics.

References

1. Kim, D., & Kim, J. (2018). Multiplayer Online Game Matchmaking with Fuzzy C-Means Clustering. Sensors, 18(6), 1967.
2. Wang, L., Chen, H., Yang, B., & Li, Y. (2021). Multiplayer Online Game Matching Method Based on Comprehensive Weight. IEEE Access, 9, 46178–46188.
3. Xu, Z., Wang, W., & Wei, X. (2019). Multiplayer Online Game Matchmaking Method Based on User Preference. In 2019 International Conference on Artificial Intelligence and Computer Engineering (ICAICE) (pp. 141–144). IEEE.
4. Lee, J., Kim, D., & Lee, D. (2017). Matchmaking System Using Bayesian Networks in Online Games. In Proceedings of the 19th ACM International Conference on Multimodal Interaction (pp. 467–471). ACM.
5. Hsieh, C. L., Hsiao, J. H., Chen, M. Y., & Chen, M. S. (2018). Adaptive Matchmaking Algorithm for Games. IEEE Transactions on Computational Intelligence and AI in Games, 10(4), 382–395.
6. Eric Peukert., & Julian Eberius. (2011). Rule-based Construction of Matching Processes
7. Lian, Y., Zeng, X., & Xiong, L. (2019). A Multiagent Q-Learning-Based Matchmaking Algorithm for Online Games. IEEE Transactions on Games, 11(3), 288–300.
8. Nieuwoudt, C. (2018). Online Game Matchmaking: An Introduction to Automated Player Grouping Algorithms. arXiv preprint arXiv:1810.08343.
9. Tan, C. T., Aung, K. M., & Lin, M. (2021). A Hierarchical Matchmaking Algorithm for Online Multiplayer Games. Applied Sciences, 11(3), 1413.
10. Salim, F., Zainuddin, Z., Jaafar, A., & Hamid, F. (2020). A Comprehensive Review on Multiplayer Online Game Matchmaking Algorithms. IEEE Access, 8, 15944–15960.