Greedy-Based Prediction of Objective-Taking Decisions in Professional Mobile Legends

Audric Yusuf Maynard Simatupang - 13524010
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: cubapetuking@gmail.com , 13524010@std.stei.itb.ac.id

***Abstract*— Mobile Legends: Bang Bang is a team-based multiplayer online battle arena game where teams must make continuous macro decisions, such as applying map pressure, taking objectives, defending structures, resetting tempo, or forcing fights. Although these decisions may vary depending on the game state and team playstyle, professional teams often show repeated decision-making patterns that can be analyzed using historical data. This paper proposes a greedy-based prediction model to predict professional team macro decisions, especially in the context of objective-taking and map-level actions. The model evaluates each possible macro decision using a weighted scoring formula consisting of transition score, team preference score, objective context score, advantage context score, structure context score, and Lord context score. The candidate decision with the highest score is selected as the predicted action. The data used in this research was manually annotated from game data of MPL Indonesia Season 17 Grand Final between BTR and ONIC. Evaluation was conducted using a rank-based scoring system on five selected game moments. The model obtained 24 out of 25 possible points, resulting in an evaluation score of 96%. This result shows that the greedy algorithm is capable of producing a simple and interpretable predictive model for professional Mobile Legends macro decision-making.**

Keywords—Mobile Legends: Bang Bang, macro decision-making, greedy algorithm, prediction model, objective-taking, professional esports

I. Introduction

Mobile Legends: Bang Bang is a five-versus-five multiplayer online battle arena game where each player performs a specific role within a team. In this game, individual decisions are closely connected to team decisions because players do not act independently from the overall strategy of the team. A player's movement, positioning, and objective-taking decisions are often influenced by the current direction of the team. However, Mobile Legends is also a highly dynamic game, where the state of the match can change quickly depending on gold difference, hero availability, map condition, objective timing, and previous engagements. As a result, different players and different teams may respond to the same situation in different ways. One team may prioritize objective control, while another team may choose to delay, pressure a different area, or avoid direct contest. This variation makes it difficult to read or predict every player's movement and every team's decision during a match.

However, professional Mobile Legends provides an interesting case for analysis. In professional matches, teams usually play with stronger coordination, clearer role distribution, and more structured macro decision-making. Although the game is still dynamic, professional teams often show recognizable tendencies in how they approach objectives, map pressure, fights, and overall control. These tendencies may differ from team to team. For example, in the M7 World Championship Grand Final, Aurora Gaming PH and Alter Ego Esports could be analyzed as teams with different approaches to macro decision-making. Aurora's playstyle can be framed as more disciplined and controlled, while Alter Ego's playstyle can be framed as more aggressive and proactive. This shows that professional teams may develop unique decision-making patterns that appear repeatedly across similar situations.

Because professional teams tend to have identifiable macro patterns, their decisions can be modeled and predicted using historical data. The goal of this paper is not to determine the objectively best decision in a given situation. Instead, this paper focuses on predicting the macro decision that a professional team is most likely to take based on previous behavior and the current game state. In other words, the prediction is based on what the team usually does, not necessarily what the team should do. This distinction is important because each team may have a different style, preference, and level of risk-taking when responding to the same game condition.

To model this decision-making process, this paper uses a greedy-based prediction approach. In simple terms, a greedy approach chooses the option that gives the highest value in the current situation without planning far ahead into future possibilities. This is suitable for modeling macro-decision prediction because each possible action can be scored based on the current game state and the team's historical tendencies. For example, the model can consider how often a team continues pressure after gaining map control, how often a team chooses to contest an objective under certain conditions, or how often a team disengages after a previous action. Each candidate macro action is given a score, and the action with the highest score is selected as the predicted decision.

Therefore, this paper proposes a greedy-based model for predicting macro decision-making in professional Mobile Legends matches. The model uses manually annotated match data containing game state information such as gold difference, hero alive status, objective timer, tower condition, Lord status, previous action, and the actual macro action taken by the team. By combining these features with a weighted scoring system, the model aims to predict the most likely macro decision a team will take in a given situation. Through this approach, the paper provides a simple and interpretable way to analyze professional team tendencies in objective-taking, map pressure, map control, and other macro-level decisions.

II. Theoretical Background

A. Mobile Legends: Bang Bang

Mobile Legends: Bang Bang is a five-versus-five multiplayer online battle arena game. In a standard match, two teams compete against each other, with each team consisting of five players. Each player controls one hero and performs a specific role within the team. The main goal of the game is to destroy the enemy base while defending the team's own base.

Although each player controls their own hero, the game is heavily team-based. A team must coordinate movements, control areas of the map, collect resources, defend structures, and take objectives in order to gain advantages. Because of this, Mobile Legends is not only determined by individual mechanical ability, but also by how well the team makes decisions together.

1) Map Layout

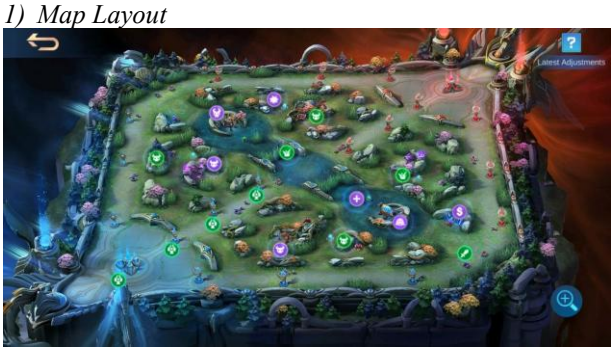

Fig. 1. Mobile Legends: Bang Bang Map Layout. Source : https://jetex.id/blog/cara-ganti-map-mobile-legends/

The Mobile Legends map consists of several important areas that influence how the game is played. In general, the map contains three lanes, jungle areas, river areas, turret structures, neutral objective areas, and the bases of both teams.

The three lanes are the upper lane, middle lane, and lower lane. The middle lane connects both bases through the central part of the map, while the upper and lower lanes are located on the sides of the map. The upper and lower lanes can function as Gold Lane or EXP Lane depending on the match configuration. These lane roles affect how players are distributed and how teams prepare their early-game movements.

Between the lanes, there are jungle areas. These areas contain neutral monsters and resources that can be taken by players. The jungle also functions as a rotation path because players can move through it to reach different lanes or objective areas. Because of this, jungle control can affect map pressure and team movement.

The river area is located around the middle of the map and connects different sides of the battlefield. Near the river area, there are neutral objective locations where major objectives such as Turtle and Lord can appear depending on the game phase. These areas are important because teams often move around them before contesting or taking objectives.

2) Objectives in Mobile Legends: Bang Bang

Mobile Legends has several major objectives that are important in determining the direction of a match. These objectives include taking Turtle, taking Lord, and destroying enemy turrets. Objectives are important because they can provide advantages such as additional resources, stronger map control, or increased pressure toward the enemy base.

Turtle is an objective that is generally more relevant in the earlier phase of the game. Taking Turtle can provide advantages that help a team strengthen its position. Lord is an objective that becomes more important in the later phase of the game because it can help a team push lanes and pressure enemy structures. Turrets are also important objectives because destroying enemy turrets opens more space on the map and makes it easier for a team to control enemy territory.

Kills, fights, rotations, and pressure are also important parts of the game, but they often function as ways to gain access to larger objectives. For example, a team may create pressure in one area to gain control over an objective area, or a team may win a fight and then use that advantage to take an objective. Therefore, objective-taking is one of the important parts of team-level decision-making in Mobile Legends.

B. Micro and Macro Decision-Making in Mobile Legends: Bang Bang

In most multiplayer online battle arena games, decision-making can generally be divided into two types: micro decision-making and macro decision-making. These two terms describe different levels of decision-making during a match. Micro decision-making focuses on individual hero control, while macro decision-making focuses on larger team-level and map-wide decisions.

1) Micro Decision-Making

Micro decision-making refers to decisions made by an individual player when controlling their hero. These decisions are usually related to mechanical execution and direct interaction with the enemy. Micro decisions are often made in short moments and require quick reactions.

Examples of micro decision-making include choosing when to use a skill, aiming an ability, dodging enemy attacks, positioning during a fight, and deciding how to trade damage against an enemy hero. These decisions usually show how well a player can control a specific hero and respond to immediate situations.

In general, micro decision-making is closely related to individual performance. A player with strong micro ability can

use their hero effectively in fights, survive dangerous situations, and execute mechanical actions accurately. However, micro decision-making alone does not fully determine the direction of a match because Mobile Legends is also influenced by larger team decisions.

2) Macro Decision-Making

Macro decision-making refers to larger decisions that involve map awareness, team movement, and overall game strategy. Unlike micro decision-making, macro decision-making is not only about controlling one hero, but also about understanding the current state of the game and deciding what the team should do as a whole.

Examples of macro decision-making include deciding which area of the map to pressure, whether to contest or give up an objective, when to rotate, when to defend, when to reset, and when to force or avoid a fight. These decisions are usually influenced by factors such as gold difference, hero availability, objective timing, turret condition, map control, and previous actions.

When macro decision-making is viewed from the team perspective, it represents the action or intention of the team as a whole. A team may decide to pressure one side of the map, prepare around an objective, defend against enemy pressure, or reset before making another move. Although each player still moves individually, their movements can form one larger team decision when they are coordinated toward the same goal.

C. Greedy Algorithm

A greedy algorithm is an algorithmic approach that chooses the most valuable option at the current step. The main idea of greedy is to make the best immediate choice based on the available information. A greedy algorithm does not examine every possible future sequence before making a decision. Instead, it focuses on selecting the option that appears to be the best in the current situation.

In a decision-making problem, a greedy algorithm usually requires a way to compare available options. This comparison is done using a value, score, or evaluation function. The option with the highest value is then selected as the greedy choice. Because of this, greedy algorithms are often simple, efficient, and easy to understand.

A greedy algorithm generally consists of several components:

1. Candidate Set

   The candidate set is the collection of all possible choices that can be selected by the algorithm. These candidates represent the available options in the current problem.

2. Solution Set

   The solution set is the collection of candidates that have been selected as part of the answer. In some greedy problems, the solution set may contain multiple selected candidates. For example, in a problem that requires several items to be selected, the solution set grows each time the algorithm chooses a candidate.

3. Solution Function

   The solution function is used to determine whether the current solution set already forms a complete solution. This function checks whether the algorithm has reached the final answer or still needs to continue selecting candidates.

4. Selection Function

   The selection function is used to choose the best candidate from the candidate set. This function determines which candidate should be selected at the current step based on a certain rule, such as the largest value, smallest cost, or highest score.

5. Feasibility Function

   The feasibility function is used to check whether a candidate can be selected without violating the constraints of the problem. A candidate may have a high value, but it still needs to be feasible according to the rules or conditions of the problem.

6. Objective Function

   The objective function defines what the algorithm tries to optimize. In a greedy algorithm, the objective function is used to measure the value of a candidate or solution. Depending on the problem, the objective may be to maximize profit, minimize cost, reduce distance, or achieve another measurable goal.

Because greedy algorithms make decisions based on the best current option, they do not always guarantee a globally optimal result for every problem. However, they are useful when the problem can be represented through clear candidates, constraints, and an evaluation rule. Greedy algorithms are also commonly used because they are straightforward to implement and often efficient compared to methods that explore all possible solutions.

III. METHODOLOGY

A. Data Preparation

1) Data Collection

The data used in this paper was collected from the MPL Indonesia Season 17 Grand Final match between BTR and ONIC. The data was manually annotated by observing the major macro actions that happened during the game. Each annotated row represents one observed game state and the macro action taken by the selected team at that moment.

The annotation focuses on team-level macro decisions rather than individual mechanical actions. Therefore, the recorded actions include decisions such as applying lane pressure, setting up or taking an objective, invading the jungle, forcing a fight, defending, resetting, or ending the game. The purpose of the dataset is to capture what a team actually does in a given game situation so that the model can learn decision tendencies from historical match data.

2) Dataset Structure

The dataset is stored in CSV format. Each row contains the current game state, team condition, objective condition, structure condition, and the macro action label. The structure of the dataset is shown in Table I.

TABLE I. DATASET STRUCTURE

Name of Data	*Value*	*Description*
match_id	Text for match code	Identifies the match and game number where the row was taken from.
game_time	Time in MM:SS format	Shows the in-game timestamp of the observed macro decision.
team	Text for team name	Shows the team whose macro action is being predicted or analyzed.
opponent	Text for team name	Shows the opposing team. The value must be different from team.
upper_lane_role	EXP, MID, or GOLD	Shows the role assigned to the upper lane in the match.
mid_lane_role	EXP, MID, or GOLD	Shows the role assigned to the middle lane. In most cases, this is MID.
lower_lane_role	EXP, MID, or GOLD	Shows the role assigned to the lower lane in the match.
team_gold	Integer	Stores the total gold of the selected team.
enemy_gold	Integer	Stores the total gold of the opposing team.
ally_jungle_alive	1 or 0	Shows whether the selected team's jungler is alive.
ally_roam_alive	1 or 0	Shows whether the selected team's roamer is alive.
ally_mid_alive	1 or 0	Shows whether the selected team's mid laner is alive.
ally_exp_alive	1 or 0	Shows whether the selected team's EXP laner is alive.
ally_gold_alive	1 or 0	Shows whether the selected team's gold laner is alive.
enemy_jungle_alive	1 or 0	Shows whether the opponent's jungler is alive.
enemy_roam_alive	1 or 0	Shows whether the opponent's roamer is alive.
enemy_mid_alive	1 or 0	Shows whether the opponent's mid laner is alive.
enemy_exp_alive	1 or 0	Shows whether the opponent's EXP laner is alive.
enemy_gold_alive	1 or 0	Shows whether the opponent's gold laner is alive.
next_objective_type	TURTLE, LORD, or NONE	Shows the next relevant neutral objective.
next_objective_timer_seconds	Integer	Shows the time until the next objective spawns. A value of 0 means the objective is available, while -1 means unknown or not applicable.
objective_side	UPPER, LOWER, NONE, or UNKNOWN	Shows the side of the map where the next neutral objective is located.
lord_status	NONE, ALLY_LORD, ENEMY_LORD, or CONTESTED	Shows the current Lord condition from the selected team's perspective.
lord_lane	UPPER, MID, LOWER, NONE, or UNKNOWN	Shows the lane where Lord is pushing, if a summoned Lord exists.
ally_upper_t1_alive to ally_lower_t3_alive	1 or 0	Shows the tower status of the selected team across upper, middle, and lower lanes. A value of 1 means the tower is still standing, while 0 means it has been destroyed.
enemy_upper_t1_alive to enemy_lower_t3_alive	1 or 0	Shows the tower status of the opponent across upper, middle, and lower lanes. A value of 1 means the tower is still standing, while 0 means it has been destroyed.
action_location	UPPER, MID, LOWER, JUNGLE, TURTLE_AREA, LORD_AREA, BASE, NONE, or UNKNOWN	Shows the main location where the macro action occurs.
current_action	One of the defined macro action labels	Shows the actual macro action taken by the selected team in the observed decision window.

In this dataset, a macro decision is represented by combining current_action and action_location. Therefore, a decision is not only defined by what the team does, but also by where the action mainly occurs. For example, a teamfight around Lord area and a teamfight in the middle lane are both categorized as fights, but they have different action locations.

3) Data Preprocessing

After the CSV data is collected, the data is processed before being used by the greedy prediction model. The preprocessing

step is implemented in the program to make the dataset easier to use for prediction.

First, the program reads each row from the CSV file and converts the required numerical columns into integer values. These numerical columns include gold values, objective timer, alive status, and tower status. The game_time column, which is written in MM:SS format, is also converted into game_time_seconds. This conversion allows the program to sort and compare rows based on numerical time instead of text format.

After the rows are read, the preprocessing step creates a copy of the dataset and sorts the rows based on match_id, team, and game_time_seconds. This sorting is important because the model uses the previous macro decision as one of the factors in prediction. By sorting the data in chronological order for each team in each match, the program can correctly identify what action happened before the current row.

The next preprocessing step is adding a decision field to each row. This field is created by combining the current_action and action_location values into one pair. In other words, the program represents each macro decision as:

decision = (current_action, action_location)

After that, the program adds a previous_decision field. For each match and team, the program stores the decision from the previous row and assigns it as the previous decision of the current row. If a row is the first recorded decision for a team in a match, the program uses a starting value:

previous_decision = (START, NONE)

Finally, the preprocessing step builds several context fields from the raw game-state values. These context fields are later used by the greedy scoring model. The generated contexts include objective context, advantage context, structure context, and Lord context. These contexts transform raw match information into grouped conditions that can be counted and compared historically.

4) Macro Action Labeling

The dataset uses a set of predefined macro action labels to represent the major team-level actions that happen during a match. Each action label is also paired with a location mark to show where the action mainly occurs, such as upper lane, middle lane, lower lane, jungle, Turtle area, Lord area, base, or no specific location. In general, a game state must fulfill at least two of the listed indicators to be labeled as a certain macro action. However, if the action has a clear hard trigger, such as directly attacking Turtle, Lord, turret, or the enemy base, the label can still be assigned when the intention is obvious. Below are the labels used and their rules:

1. PRESSURE_UPPER, PRESSURE_MID, PRESSURE LOWER

   These labels are used when the team tries to gain control of a specific lane or nearby map area. This label is assigned when at least two of the following indicators are fulfilled:

   - The team pushes or clears the lane wave with pressure or rotation intention.
   - Two or more team members move toward or gather around that lane or nearby river area.
   - The enemy is forced to retreat, clear the wave, or defend that side.
   - The team controls nearby bush, river, or jungle entrance.
   - The action prepares for a turret, objective, invade, gank, or fight.

2. TURTLE SETUP, LORD SETUP

   These labels are used when the team plays around an objective area without fully committing to taking the objective. This label is assigned when at least two of the following indicators are fulfilled:

   - The team controls the river, bush, or entrance around the objective area.
   - The team positions around the objective without fully committing.
   - The team briefly hits the objective as bait but does not seriously reduce its HP.
   - The team zones the enemy away from or around the objective area.
   - The team waits near the objective to force enemy movement or contest.

3. TAKE TURTLE, TAKE LORD

   These labels are used when the team clearly commits to contesting or securing a major neutral objective. This label is assigned when at least two of the following indicators are fulfilled:

   - The team starts seriously attacking the objective.
   - The objective HP is intentionally being reduced.
   - The jungler or objective taker is positioned to secure the objective.
   - The team continues focusing the objective despite possible enemy contest.
   - The team wins nearby pressure or a fight and immediately attempts the objective.

4. TAKE_UPPER_TURRET, TAKE_MID_TURRET, TAKE LOWER TURRET

   These labels are used when the team commits to sieging or destroying a turret in a specific lane. This label is assigned when at least two of the following indicators are fulfilled:

   - The team directly attacks the turret.
   - The team groups or positions to siege the turret.

- The enemy is forced away from defending the turret.
- The minion wave is used to pressure the turret.
- The turret becomes the main visible target of the action.

5. INVADE JUNGLE

   This label is used when the team enters the enemy jungle to steal resources, deny farming space, or control jungle territory. This label is assigned when at least two of the following indicators are fulfilled:

   - The team enters the enemy jungle area.
   - The team steals or attempts to steal enemy jungle camps or buffs.
   - The team zones the enemy away from their own jungle.
   - Two or more team members control enemy jungle space.
   - The team uses the invade to deny farming routes or create map pressure.

6. TEAMFIGHT SKIRMISH

   This label is used when a multi-hero fight becomes the main action. This label is assigned when at least two of the following indicators are fulfilled:

   - Multiple heroes from both teams are involved.
   - Both teams commit skills, ultimates, or major resources.
   - The fight affects objective control, lane control, or map pressure.
   - The action shifts from setup, pickoff, or objective-taking into direct fighting.
   - The fight becomes the main focus of the current decision window.

7. PICKOFF GANK

   This label is used when the team attempts to catch or kill one isolated enemy hero. This label is assigned when at least two of the following indicators are fulfilled:

   - One enemy hero is isolated or out of position.
   - The team collapses specifically onto that target.
   - The action is focused on killing or forcing that target away.
   - The fight does not immediately become a full multi-hero team fight.
   - The action is carried out by a small group, usually one to three heroes.

8. RESET RECALL

   This label is used when the team disengages, recalls, or resets tempo. This label is assigned when at least two of the following indicators are fulfilled:

   - Multiple team members recall or move back to safer positions.
   - The team stops pressuring after an objective, fight, or push.
   - The team disengages instead of continuing the contest.
   - The team resets to heal, buy items, or prepare the next wave or objective.
   - The team avoids further combat despite having been near the enemy or objective.

9. DEFEND CLEAR

   This label is used when the team's main action is defensive. This label is assigned when at least two of the following indicators are fulfilled:

   - The team clears incoming enemy minion waves.
   - The team protects a turret or base structure.
   - The team responds to enemy lane pressure.
   - The team defends against Lord push.
   - The team is forced backward and plays to stabilize the map.

10. END GAME

    This label is used when the team commits to ending the game by attacking the enemy base. This label is assigned when at least two of the following indicators are fulfilled:

    - The team directly attacks the enemy base or crystal.
    - The team ignores other objectives and focuses on finishing the game.
    - The enemy base is exposed or nearly exposed.
    - Several enemy heroes are dead or unable to defend.
    - The team continues hitting base structures instead of resetting or taking other objectives.

B. Greedy Prediction Model

1) Greedy Component Definition

The greedy components in this prediction model are defined as follows:

1. Candidate Set

   In this research, each candidate represents one possible team-level action and the location where the action occurs. The candidate set defines the possible outputs of the prediction model.

2. Solution Set

   Since the purpose of this model is to predict the next macro decision, the final solution consists of the selected macro decision with the highest score.

3. Solution Function

   In this research, the solution is considered complete when the candidate decisions have been evaluated and the best candidate has been selected as the prediction.

4. Selection Function

   In this research, each candidate is evaluated using a scoring function, and the candidate with the highest final score is selected.

5. Feasibility Function

   In this research, a candidate is feasible if its action and location are compatible with the defined macro action rules.

6. Objective Function

   In this research, the objective is to maximize the prediction score of a candidate macro decision. The candidate with the highest score is considered the most likely decision based on the given game state and historical decision patterns.

2) Scoring Formula

Each candidate decision is evaluated using a weighted scoring formula. The final score combines several score components, where each component represents a different factor that may influence the team's macro decision. The formula used in the model is:

$$\text{Score}(d, s) =$$
$$0.30 \times \text{TransitionScore}(d, s) +$$
$$0.15 \times \text{TeamPreferenceScore}(d) +$$
$$0.20 \times \text{ObjectiveContextScore}(d, s) +$$
$$0.15 \times \text{AdvantageContextScore}(d, s) +$$
$$0.10 \times \text{StructureContextScore}(d, s) +$$
$$0.10 \times \text{LordContextScore}(d, s)$$

Where:

d = candidate macro decision

s = current game state

The total weight is equal to 1.00, meaning that every component contributes to the final score according to its assigned weight. The largest weight is given to the transition score because the previous macro decision is considered an important indicator of the next possible decision. Each component in the formula is calculated as follows.

1. TransitionScore

   TransitionScore measures how often a candidate decision occurs after the previous macro decision. This component represents the historical sequence pattern between one macro decision and the next macro decision.

$$\text{TransitionScore}(d, s) =$$
$$(\text{count}(\text{previous\_decision} \rightarrow d) + 1) /$$
$$(\text{count}(\text{previous\_decision} \rightarrow \text{any\_decision}) + N)$$

   Where:

   previous_decision = macro decision that happened before the current state

   d = candidate macro decision

   N = total number of candidate decisions

2. TeamPreferenceScore

   TeamPreferenceScore measures how often a team chooses a candidate decision in general. This component represents the overall macro tendency of the team regardless of the previous decision or current game context.

$$\text{TeamPreferenceScore}(d) =$$
$$(\text{count}(d) + 1) / (\text{total\_decisions} + N)$$

   Where:

   d = candidate macro decision

   total_decisions = total number of macro decisions recorded for the team

   N = total number of candidate decisions

3. ObjectiveContextScore

   ObjectiveContextScore measures how often a candidate decision occurs under a similar neutral objective condition. This component uses the objective context, which consists of the next objective type, objective timer category, and objective side.

$$\text{ObjectiveContextScore}(d, s) =$$
$$(\text{count}(\text{objective\_context} \rightarrow d) + 1) /$$
$$(\text{count}(\text{objective\_context} \rightarrow \text{any\_decision}) + N)$$

   Where:

   objective_context = (next_objective_type, objective_timer_bucket, objective_side)

   d = candidate macro decision

   N = total number of candidate decisions

4. AdvantageContextScore

   AdvantageContextScore measures how often a candidate decision occurs under a similar advantage condition. This component uses the advantage context, which consists of gold difference, alive hero difference, and jungler state.

$$\text{AdvantageContextScore}(d, s) =$$
$$(\text{count}(\text{advantage\_context} \rightarrow d) + 1) /$$

(count(advantage_context → any_decision) + N)

Where:

advantage_context = (gold_bucket, alive_bucket, jungler_state)

d = candidate macro decision

N = total number of candidate decisions

5. StructureContextScore

   StructureContextScore measures how often a candidate decision occurs under a similar tower condition. This component uses the structure context, which represents the number of destroyed towers by tier for both teams.

   StructureContextScore(d, s) =

   (count(structure_context → d) + 1) /

   (count(structure_context → any_decision) + N)

   Where:

   structure_context = (enemy_outer_destroyed_count, enemy_inner_destroyed_count, enemy_base_destroyed_count, ally_outer_destroyed_count, ally_inner_destroyed_count, ally_base_destroyed_count)

   d = candidate macro decision

   N = total number of candidate decisions

6. LordContextScore

   LordContextScore measures how often a candidate decision occurs under a similar Lord condition. This component uses the Lord context, which consists of Lord status and Lord lane.

   LordContextScore(d, s) =

   (count(lord_context → d) + 1) /

   (count(lord_context → any_decision) + N)

   Where:

   lord_context = (lord_status, lord_lane)

   d = candidate macro decision

   N = total number of candidate decisions

3) Prediction Procedure

The prediction procedure begins by loading the historical dataset from the CSV file. The rows are then preprocessed so that each row contains the macro decision, previous decision, and the context fields required by the scoring model. After preprocessing, the greedy scorer is trained using the historical rows by counting team preferences, transition patterns, and context-based decision frequencies.

When predicting a new game state, the program asks for the current match condition, such as the selected team, opponent, lane role setup, gold values, alive status, objective information, tower state, and previous macro decision. If the previous macro decision is unknown, the program uses a starting value to represent the beginning of the decision sequence.

After the current game state is entered, the program builds the same context fields used in the historical data. The model then generates all valid candidate macro decisions from the predefined action-location combinations. Each candidate is scored using the weighted scoring formula. The candidates are sorted from the highest final score to the lowest final score.

The program outputs the top five predicted macro decisions. Each prediction contains the predicted action, action location, final score, and score breakdown for each component. The highest-ranked prediction is treated as the main greedy prediction, while the other four predictions are used as alternative possible decisions.

4) Evaluation Procedure

The evaluation is performed by testing the prediction model on several selected game moments. In this evaluation, five moments are taken from a match and used as test cases.

For each test case, the game state is given as input to the prediction model. The model then calculates the score of every possible macro decision and returns the five highest-scoring predictions. The predicted results are then compared with the actual macro action that happened in the match.

Since the model produces five ranked predictions instead of only one prediction, the evaluation uses a rank-based scoring system. A higher score is given when the actual macro action appears at a higher prediction rank. The actual macro action receives a maximum of 5 points when it appears as the top prediction, with the score decreasing by 1 point for each lower rank. If the actual action does not appear among the five predicted actions, no points are awarded.

The total score is calculated by adding the score obtained from all test cases. Since five test cases are used and the maximum score for each test case is 5 points, the maximum possible score is:

Maximum Score = 5 × 5 = 25

The final evaluation percentage is calculated using the following formula:

Evaluation Score =

(Total Obtained Score / Maximum Score) × 100%

The result of this evaluation is used to show how closely the greedy prediction model can predict the actual macro decisions made by the team in selected game situations.

IV. Evaluation and Discussion

A. Evaluation

The evaluation was conducted using Game 1 of the MPL Indonesia Season 17 Grand Final between BTR and ONIC. The evaluation results are shown in Table II.

TABLE II. Evaluation results

Test Case	Game Time	Team	Top-5 Predictions	Actual Action	Score
1	2:07	BTR	TURTLE SETUP at TURTLE AREA, TAKE TURTLE at TURTLE AREA, TEAMFIGHT SKIRMISH at TURTLE AREA, PRESSURE MID at MID, RESET RECALL at NONE	TURTLE SETUP at TURTLE AREA	5
2	4:42	BTR	TAKE TURTLE at TURTLE AREA, TURTLE SETUP at TURTLE AREA, TEAMFIGHT SKIRMISH at TURTLE AREA, PRESSURE MID at MID, RESET RECALL at NONE	TAKE TURTLE at TURTLE AREA	5
3	5:44	BTR	RESET RECALL at NONE, INVADE JUNGLE at JUNGLE, TAKE MID TURRET at MID, TAKE LOWER TURRET at LOWER, TAKE UPPER TURRET at UPPER	INVADE JUNGLE at JUNGLE	4
4	6:07	BTR	TEAMFIGHT SKIRMISH at LOWER, TAKE LOWER TURRET at LOWER, PRESSURE LOWER at LOWER, TAKE TURTLE at TURTLE AREA, PRESSURE MID at MID	TEAMFIGHT SKIRMISH at LOWER	5
5	10:48	BTR	TAKE LOWER TURRET at LOWER, PRESSURE LOWER at LOWER, PRESSURE MID at MID, RESET RECALL at NONE, TAKE MID TURRET at MID	TAKE LOWER TURRET at LOWER	5

Based on the evaluation results, the model obtained a total score of 24 out of the maximum possible score of 25. Using the rank-based evaluation formula, the final evaluation score is:

$$\text{Evaluation Score} = (24 / 25) \times 100\% = 96\%$$

Therefore, the greedy prediction model achieved a final evaluation score of 96% on the selected test cases.

B. Discussion

Based on the evaluation result, the greedy prediction model obtained 24 out of 25 possible points, resulting in an evaluation score of 96%. This result shows that the model was able to place the actual macro decision within the top-ranked predictions for most of the selected test cases. The high score suggests that the combination of historical transition patterns, team preference, objective context, advantage context, structure context, and Lord context can represent the team's macro-decision tendency reasonably well.

However, this result should not be interpreted as a complete measurement of the model's performance in all situations. The evaluation was conducted using a limited number of selected moments from Game 1 of the MPL Indonesia Season 17 Grand Final between BTR and ONIC. Because of this, the result may still be affected by the small number of test cases and the specific characteristics of the match being evaluated. In addition, the dataset was manually annotated, so the consistency of action labeling depends on the clarity of the annotation rules and the accuracy of the observation.

For future work, the model can be improved by using a larger dataset that includes more games, more teams, and more tournament stages. A larger evaluation set would make the performance measurement more reliable. The model can also be compared with other prediction approaches to evaluate whether the greedy method performs better or worse than alternative methods. Furthermore, additional game-state features, more detailed action labels, or improved annotation procedures may help the model capture macro-decision patterns more accurately.

V. Conclusion

This research shows that the greedy algorithm is capable of producing a good predictive model for professional Mobile Legends: Bang Bang macro decision-making, especially in the context of objective-taking and team-level map decisions. By assigning scores to each possible macro action and selecting the highest-scoring candidate, the model can predict the decision that a team is most likely to take based on historical patterns and the current game state.

The evaluation result supports this conclusion, as the model obtained 24 out of 25 possible points, resulting in an evaluation score of 96%. This shows that the model was able to place the actual macro decision within the top-ranked predictions for most of the selected test cases.

Therefore, the greedy-based approach can be considered suitable for this prediction problem because it is simple, interpretable, and able to represent professional team decision tendencies. However, the model can still be improved in future work by using a larger dataset, adding more matches and teams, and comparing its performance with other prediction methods.

VI. Appendix

Video Link at Youtube

https://youtu.be/RILzInZn_Dw

Program Link at GitHub

https://github.com/alivovue/Makalah_Stima

Acknowledgment

First and foremost, the author would like to express gratitude to God Almighty for His blessings and guidance, which enabled the completion of this paper.

The author would also like to express sincere appreciation to Ir. Rila Mandala, M.Eng., Ph.D., the lecturer of K02 IF2211 Strategi Algoritma, for the knowledge, guidance, and learning opportunities provided throughout the course.

Lastly, the author would like to congratulate the champion of MPL Indonesia Season 17 for their achievement and for providing an inspiring professional match that became part of the motivation and context for this paper.

References

[1] MPL Indonesia, "LIVE | MPL ID S17 | Playoffs Grand Finals | Bahasa Indonesia," YouTube. [Online]. Available: https://www.youtube.com/watch?v=u7dLw-qpMMM&t=13732s. Accessed: Jun. 19, 2026.

[2] R. Munir, "Algoritma Greedy (Bagian 1)," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, Institut Teknologi Bandung, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/04-Algoritma-Greedy-(2026)-Bag1.pdf. Accessed: Jun. 19, 2026.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

Audric Yusuf Maynard Simatupang