Automated Unit Balancing in Tactical RPGs via Scenario Based Simulation and Heuristic Search

Rafizan Muhammad Syawalazmi - 13523034
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: rafi.syawalazmi@gmail.com , 13523034@std.stei.itb.ac.id

***Abstract*—Balancing unit statistics in tactical role playing games (TRPGs) is a critical but notoriously complex task, often constrained by the limitations of manual playtesting and subjective developer intuition. This paper presents an automated framework for unit balancing based on heuristic search and scenario-based evaluation. Units are modeled using core combat attributes and are tested against a diverse set of predefined combat scenarios to simulate near optimal play. The simulation engine employs a depth limited depth-first search (DFS) with alpha-beta pruning and top N action filtering to explore plausible combat outcomes efficiently. A win rate-based fitness function evaluates unit performance across all scenarios, offering an objective and quantifiable balance metric. A random search strategy is used to tune unit parameters until their aggregated performance matches a defined baseline. The resulting system reduces the subjectivity and time cost of manual balancing, offering a scalable, extensible, and data-driven alternative for developing fair and strategically deep tactical game experiences.**

Keywords—Game balancing, tactical RPG, scenario-based evaluation, DFS, alpha-beta pruning, random search, AI simulation, unit optimization

I. INTRODUCTION

Game balance is an important piece of successful game design, fundamentally influencing player retention, satisfaction, and a game's long term viability. An unbalanced game, whether perceived as extremely difficult or easy, leads to player frustration or boredom, often resulting in player disengagement. The objective of balancing is to create a fair and enjoyable experience, offering a sense of accomplishment when challenges are overcome, without falling into overwhelming difficulty or a lack of meaningful opposition.

However, achieving this equilibrium through traditional, manual methods is with significant limitations. Modern games are characterized by their depth and their vast global player base, rendering it practically impossible for even large development teams to fully comprehend every conceivable combination or variation of gameplay. Manual balancing processes rely heavily on iterative human playtesting, a method that is extremely time consuming and often insufficient for games featuring complex, intransitive mechanics. This often necessitates the release of numerous balance patches post-launch, indicating an ongoing struggle to achieve stability. Furthermore, balance decisions made purely on numerical data or solely on subjective feeling often lead to unwelcomed outcomes.

The manual balancing process is also quite subjective, heavily dependent on developer intuition and the limited scope of internal playtesting and player feedback. The cognitive load imposed on a single designer or even a dedicated team to predict all tactical implications of stat modifications in a complex grid based game is immense. For example, in games like Fire Emblem Fates Conquest, enemy AI is designed to effectively target weaker units, and resource management is critical, making precise manual tuning of unit capabilities quite challenging. This human cognitive limitation means that despite best efforts, manual balancing can lead to unintended dominant strategies, situations where decisions become meaningless, or a general lack of counterplay, diminishing the strategic depth of the game.

In response to the inherent limitations of balancing, specifically manual unit balancing in grid based tactical games for this paper, characterized by its time consuming nature and subjective outcomes, this project proposes a systematic and automated solution. The core objective is to automatically balance a game unit by evaluating its performance across a predefined set of diverse combat scenarios.

Using unit modeling, an evaluator, various scenario sets, a fitness function, and a balancing strategy, creates a powerful synergy for great evaluation. The near optimal play can be achieved by the DFS-based engine. Furthermore, the use of a diverse set of predefined scenarios ensures that this competent evaluation is more flexible and does not limits the balance for specific scenarios. This prevents the unit from being inadvertently balanced for one specific enemy type while remaining overpowered or underpowered against others. This synergistic combination renders the evaluation not only computationally efficient but also highly reliable and comprehensive. It directly addresses the inherent subjectivity of manual balancing by providing objective, data-driven performance metrics under realistic and challenging conditions, thereby yielding a more robust and generalizable balance configuration.

II. THEORETICAL BASIS

A. *Depth First Search*

Depth First Search (DFS) is a fundamental graph traversal algorithm that forms the basis for exploring complex structures like game trees in artificial intelligence. It operates by systematically exploring as far as possible along each branch before backtracking. When traversing a graph, DFS explores an adjacent vertex and then recursively completes the traversal of all vertices reachable through that adjacent vertex before moving to other neighbors. This behavior is similar to a tree traversal where an entire subtree (example, the left subtree) is fully explored before moving to another (example, the right subtree).

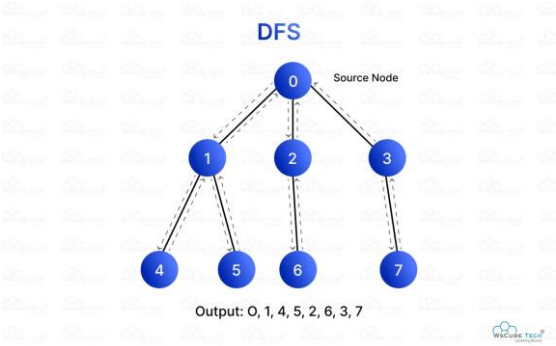

Figure 2.1. DFS Visualization. Source: https://www.wscubetech.com/resources/dsa/dfs-vs-bfs

A key consideration in graph traversal, particularly for graphs that may contain cycles, is to prevent redundant processing of nodes. DFS addresses this by typically employing a mechanism, such as a boolean visited array, to keep track of already explored nodes. The algorithm can be implemented to start from a given source node and explore all reachable vertices, or to perform a complete traversal of a disconnected graph by iteratively calling the single-source DFS for all unvisited vertices. The time complexity for DFS is generally O(V + E), where V represents the number of vertices and E represents the number of edges in the graph. Its auxiliary space complexity is also O(V + E), primarily due to the visited array and the call stack for recursive operations.[1]

In the context of game AI, DFS provides the underlying search mechanism for algorithms like Minimax and Alpha Beta pruning, which explore the vast game tree to determine optimal moves. Its depth first nature allows for a thorough examination of potential move sequences and their consequences, making it a favored approach for game playing programs due to its simplicity and effectiveness in traversing deep branches of the game tree.[2]

B. *Minimax*

The simulation engine uses a search typically used for competitive environments with conflicting goals. The Minimax algorithm is foundational for optimal decision making, constructing and traversing a game tree of possible moves and states. It recursively evaluates nodes, maximizing the current player's utility while minimizing the opponent's, assuming optimal counter play. This framework ensures the unit's performance is assessed against a challenging, strategic opponent, crucial for producing robust balance configurations. Aiming for near optimal play ensures the win-rate metric reliably indicates balance against a competent player, not a naive one.

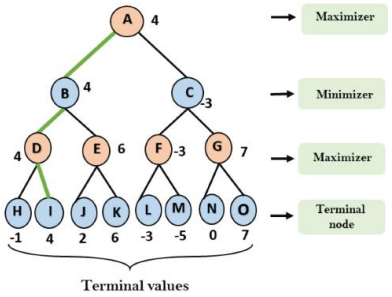

Figure 2.2. Minimax Visualization. Source: https://medium.com/@aidenrtracy/the-minimax-algorithm-f6e8e0a1eadb

There is a pruning method called Alpha Beta pruning that optimizes Minimax by reducing computation time through intelligent branch elimination. It maintains alpha (best value for maximizer) and beta (best value for minimizer) values. if alpha is more than beta, the branch is pruned. Alpha Beta is favored for its efficiency in game playing programs and can be enhanced by transposition tables and narrow search windows.

This paper also uses a "Top-N move" pruning method that complements Alpha-Beta by focusing the search on promising moves, balancing search depth for near optimal play with computational time for numerous simulations.

C. *Tactical Role Playing Games*

Tactical Role Playing Games (TRPGs) are defined by their unique fusion of deep character development and strategic combat, typically unfolding on grid based maps. These games emphasize strategic decision-making, often presenting battles as intricate puzzles where players must skillfully outmaneuver their adversaries. The genre is characterized by grid-based movement, often on hexagonal maps, and a turn-based combat system where players control their entire party's actions before the enemy's turn commences. This 'full team turns' mechanic is a defining feature, allowing for comprehensive strategic planning within each round.

The strategic depth of these games is enriched by several key mechanics. Positioning is paramount, as controlling terrain, gaining initiative, and utilizing environmental cover can significantly influence battle outcomes. Features such as zones of control restrict the movement of engaged units, while flanking or rear attacks offer tactical advantages, potentially negating enemy counterattacks or bypassing defensive measures like block points. The inclusion of diverse attack types, including ranged attacks from archers, line attacks that strike multiple units in a row, and wide area attacks from two handed weapons, adds layers of tactical variety. Furthermore, support units like healers and a wide array of active and passive skills, offering buffs to allies or debuffs to enemies,

significantly expand the tactical options available to players. Missions often incorporate secondary objectives beyond simple enemy elimination, adding further complexity and strategic considerations.

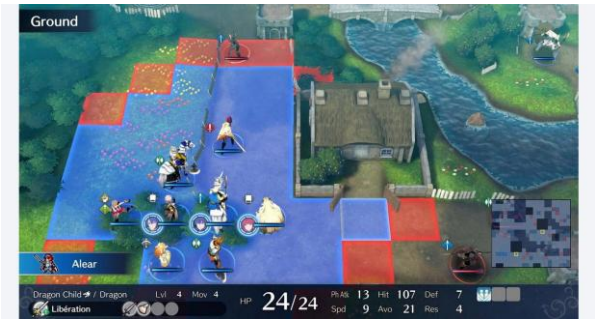

Figure 2.1. TRPG Fire Emblem. Source: https://www.gamestop.com/video-games/nintendo-switch/products/fire-emblem-engage/20001849.html

The effectiveness of a unit's design is deeply intertwined with the tactical environment provided by the map. Mechanics such as zones of control, flanking or rear attacks, and block points are heavily influenced by a unit's position on the grid-based maps. Similarly, the importance of positioning and the impact of terrain effects are repeatedly emphasized within the genre. This inherent interdependence implies that true unit balance is not merely a matter of internal numerical consistency but also how a unit performs across a variety of tactical environments. A unit perfectly balanced for open terrain might be overpowered in tight corridors or underpowered if its range cannot be effectively leveraged due to obstacles. The project's use of a diverse set of predefined scenarios implicitly addresses this, as these scenarios likely represent different map layouts or enemy compositions that challenge units in varied ways, thereby fostering a more robust balancing outcome.

In this project, game units are modeled using a set of core attributes that are fundamental to their performance in turn based and grid based combat. These attributes are standard across the tactical RPG genre and include:

- HP (Health Points): This attribute determines the total health of a character, directly impacting its survivability and endurance in combat encounters.
- ATK (Attack): This statistic dictates the damage dealt by a unit's equipped weapons, serving as its primary offensive capability.
- DEF (Defense): This attribute reduces the amount of damage taken from enemy attacks, contributing to a unit's overall resilience and durability.
- MOV (Movement): Representing the number of grid spaces a unit can traverse per turn, this attribute is critical for tactical positioning, flanking maneuvers, and controlling the battlefield.
- RNG (Range): This attribute defines the maximum distance from which a unit can initiate an attack. Its importance is implied by the presence of ranged attacks from archers in tactical games.
- SPD (Speed): This attribute influences a unit's evasion (dodge chance), its priority in counterattacks, and its ability to perform double attacks if its speed significantly exceeds that of an opponent (example by 4 points).
- CRIT (Critical Chance): This attribute boosts a unit's accuracy and hit chance, determining the likelihood of successfully landing an attack and potentially dealing critical damage.

These core attributes combine to form derived combat statistics, such as a unit's overall Attack value, which might combine its Strength with the damage rating of its equipped weapon. This intricate interplay of statistics creates complex interactions, such as the Speed versus Speed comparison for double attacks, which underscores the profound challenge of manual balancing.

The impact of changing a single unit statistic is often not linear but rather threshold-based or multiplicative. For example, a small increment in a unit's Speed might have no observable effect until it crosses a specific threshold relative to an opponent's Speed, at which point it suddenly grants a significant advantage, such as a double attack. Similarly, accuracy might scale non-linear with the Skill attribute. This non linear behavior makes manual balancing exceptionally difficult, as minor numerical tweaks can lead to disproportionately large effects, or conversely, no effect at all, until a critical breakpoint is reached. Automated simulation and optimization are therefore indispensable for effectively navigating this complex, non linear performance landscape and achieving precise unit balance.

III. IMPLEMENTATIONS

A. Unit and Battlefield Representation

Each unit in the system is modeled as a structured object with attributes such as maximum hit points (HP), attack power (ATK), defense (DEF), movement range (MOV), attack range (RNG), speed (SPD), and critical hit rate (CRIT). These attributes determine how a unit performs in combat. A unit is also associated with a team designation, either red or blue, and is assigned a fixed position on a 2D grid-based battlefield. The battlefield itself is represented as a finite-sized grid that manages tile-based positioning and enforces constraints on unit movement, such as staying within bounds and avoiding occupied tiles. This representation mirrors the structure of typical turn-based tactical games and provides a deterministic foundation for simulating discrete combat scenarios.

B. Battle Simulations

Combat simulation is implemented using a depth limited recursive search inspired by the Minimax algorithm. Each layer of recursion corresponds not to a single unit's action, but to a full team turn meaning that all remaining units on a team execute their actions before the turn passes to the opposing side. This model more accurately reflects the turn structure

found in tactical games like Fire Emblem, where players control multiple units per phase. The system explores action sequences using depthnfirst search, but integrates alpha beta pruning to eliminate branches that cannot influence the final outcome. To further control the branching factor, the system filters actions using a top N heuristic. For each unit, the simulation evaluates all legal move and attack combinations, ranks them based on factors such as potential damage and proximity to enemies, and selects only the highest scoring N actions for further evaluation. This selective pruning significantly improves runtime without discarding the most promising tactical options. The recursion continues until the maximum depth is reached or until one team has been completely defeated.

C. *Heuristic Evaluation*

At the leaves of the search tree, or when the recursion depth limit is reached, the system evaluates the resulting game state using a heuristic function. The primary metric is the difference in total remaining hit points between the red and blue teams. A positive score indicates a favorable state for the red team, while a negative score favors the blue team. This continuous scoring mechanism provides gradient feedback for optimization and allows the balancer to compare outcomes across scenarios.

In addition to scoring the outcomes, the simulator also records the sequence of actions that led to each evaluated state. This is done using an ActionStep structure, which captures the acting unit, the origin and destination of its movement, the type of action performed (e.g., move, attack, or move attack), and the target unit if applicable. As the recursive search proceeds, the optimal path is propagated upward along with the score, making it possible to extract the best move sequence under the assumed depth and search conditions.

D. *Scenario Based Evaluation*

To evaluate unit effectiveness in a balanced and generalizable way, the system tests each candidate configuration against a fixed set of combat scenarios. Each scenario defines an enemy unit with specific stats and a starting position, intended to represent different combat archetypes such as ranged attackers, high-defense tanks, or fast melee units. These scenarios remain constant across all candidate evaluations to ensure consistency.

For each scenario, the candidate unit is paired against the scenario enemy on a fresh battlefield, and the battle is simulated using the heuristic DFS engine. The outcome is recorded as a win, loss, or draw based on the evaluation score, and a binary win-rate score is computed for the entire set. This scenario-based approach provides a more comprehensive picture of unit viability than single-match evaluation and encourages general robustness rather than optimization for narrow cases.

E. *Random Stat Optimization*

The auto-balancing mechanism employs a random search strategy over a predefined space of tunable parameters. Each candidate configuration is generated by sampling values for selected stats such as HP, ATK, and DEF within bounded integer ranges. Other stats, such as movement range or critical rate, may be held constant to simplify the search.

Each candidate configuration is evaluated using the same scenario-based performance function as described above. The system computes the total win rate across all scenarios and compares this value against a target, either a fixed percentage (e.g., 50%) or the known win rate of a baseline reference unit. The configuration that minimizes the absolute difference between the candidate and target win rates is retained as the best-performing unit. This process is repeated for a fixed number of iterations (typically several hundred) to ensure adequate coverage of the search space.

IV. Testing And Analysis

Testing is done using these configurations as the basis.

Random Trials	500
Depth per Simulation	3
Top-N Actions	3
Scenario Set	One versus one against 6 combat archetypes.
Reference Unit	BaselineKnight

Table 4.1. Simulation configurations.

BaselineKnight, being as the reference unit for the simulation, provides these results when put inside the 6 different scenarios.

```
Scenario Results for BaselineKnight
Scenario   Win   Loss  Score
Archer     V           +14
Rogue      V           +10
Tank             V     -10
Mage       V           +5
Brawler          V     -5
Spearman   V           +2
Total WR: 4/6 (66.7%)
```

Figure 4.1. BaselineKnight results.

Using these win rate, the program will create a candidate that has similar win ratio. These are two candidate results using different results:

```
Scenario Results for CandidateKnight
Scenario   Win   Loss  Score
Archer     V           +18
Rogue      V           +9
Tank             V     -14
Mage       V           +6
Brawler          V     -7
Spearman   V           +3
Total WR: 4/6 (66.7%)
```

Figure 4.2. CandidateKnight results.

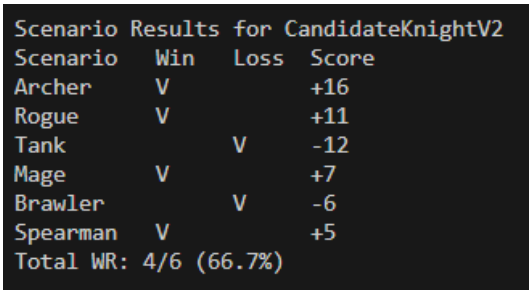

Figure 4.3. CandidateKnightV2 results.

As we can see, both candidates got the exact same win rates among the 6 scenarios provided. Both having the same win and losses for every scenarios is entirely coincidental. This behavior will disappear when using a vast more scenarios with more units in each of them. Also something to notice is that the score of each scenarios is somewhat similar with a margin below 5.

And lastly, these are the optimized and 'most balanced' configurations based on each candidates.

Figure 4.4. CandidateKnight optimized config.

Figure 4.5. CandidateKnightV2 optimized config.

V. Conclusion

This paper highlights the increasing challenges of manually balancing units in complex, grid based tactical games. The combinatorial complexity of game elements makes traditional methods time consuming, subjective, and insufficient, often leading to frequent post-release patches. This bottleneck in human cognitive capacity necessitates automated solutions.

The paper offers an automated unit balancing system driven by scenario based simulation. It leverages adversarial search algorithms (Minimax with DFS and Alpha Beta pruning) to simulate near optimal play, ensuring meaningful win rate evaluation against competent opponents. A diverse set of predefined combat scenarios ensures robust and generalizable balance.

The win rate based fitness function provides an objective, quantifiable measure for balance, complementing manual tuning's subjectivity. Random search is a pragmatic optimization strategy, offering simplicity and effectiveness in exploring the high-dimensional, non-linear parameter space of unit statistics.

In essence, this research demonstrates a viable path towards more efficient, objective, and robust game balancing. By automating complex numerical optimization, game developers can potentially reduce development time, minimize post release issues, and deliver more consistently enjoyable and strategically deep tactical games.

Acknowledgment

The Author would like to express our heartfelt gratitude to everyone who contributed to the successful completion of the paper.

The Author extend his deepest appreciation to his algorithmatic strategies teacher, Dr. Nur Ulfa Maulidevi, for their teachings to make this paper possible and encouraging us to make this paper in the first place.

This paper is collaborative effort and shared knowledge of all those involved. Thank you.

References

[1] Adversarial Search Algorithms in Artificial Intelligence (accessed June 24, 2025, https://www.geeksforgeeks.org/artificial-intelligence/adversarial-search-algorithms/

[2] Best-First and Depth-First Minimax Search in Practice, accessed June 24, 2025, https://webdocs.cs.ualberta.ca/~jonathan/publications/ai_publications/csn.pdf

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, 24 Juni 2025

Rafizan Muhammad Syawalazmi (13523034)