Hybrid Genetic Programming and Q-Learning for Behavior Tree Evolution in MicroRTS Tactical Scenarios

Mohamed Amine Chikh Touami[1], Mohammed Salem[1], Mohamed Fayçal Khelfi[2]
[1]LISYS Lab, University of Mascara, Algeria
[2]ESGEE Oran, Algeria
E-mail: ma.chikhtouami@gmail.com, mf_khelfi@yahoo.fr

Keywords: Behavior trees, genetic programming, reinforcement learning, Q-learning, real-time strategy games, MicroRTS, hybrid soft computing

Received: April 6, 2026

Genetic Programming (GP) can be used to evolve human-interpretable Behavior Tree controllers for real-time strategy games. Current Behavior Trees-based GP approaches compute fitness only at the end of each episode, which does not allow learning from finer-grained tactical decisions during gameplay. In this work, we integrate tabular Q-learning within the BT controller to gate actions at terminal nodes based on learned action-values, and collect additional rewards at each tick of the game that are used to augment the final fitness signal. Since the Q-table is reset at the start of each episode, our approach allows the learned information to affect fitness, while keeping the learned values from one generation separate from subsequent generations (a form of Baldwinian learning). Applied to plain-terrain MicroRTS micromanagement challenge (population of 100 BTs, 2000 generations, against a deterministic rush opponent), our approach achieves a maximum fitness of 26.44 compared to GP's max fitness of 22.0, a relative improvement of 20.2%. Mean best-fitness was also increased by 17.8% compared to the GP-only baseline. These results are reported from a single representative run per configuration; multi-seed replication is identified as a priority for future work. The additional per-episode RL reward signal is strongly correlated with elite fitness ($r = 0.89$), confirming that it provides informative guidance for the evaluation. Unit coordination exhibited by the hybrid agents is also found to be more structured, with defined front-line and support roles. The agent maintains its interpretable Behavior Tree structure.

Povzetek: Prispevek predstavlja hibridni pristop, ki združuje genetsko programiranje in Q-učenje za evolucijo vedenjskih dreves v strateških igrah v realnem času, dosegajoč boljše rezultate pri ohranjanju interpretabilnosti.

1 Introduction

Real-Time Strategy (RTS) games are a challenging domain for artificial intelligence research. Computer agents must operate under severe time pressure, uncertainty, high-dimensional state and action spaces and have to face increasingly difficult adversaries. Having a suitable benchmark for the evaluation of single RTS game-playing AI agents is crucial to compare the advances and performances of different approaches. However, common benchmarks such as a competitive multiplayer version of Starcraft2, are very hard to replicate, due to their complexity and scale. Commercial RTS games such as Warcraft III, Starcraft or Age of Empires II are hard to scale down for empirical evaluations because the amount of knowledge required to make them playable increases exponentially with the reduction of their inherent complexity. The MicroRTS [21] game is a downscaled RTS benchmark that was designed specifically to be easy to understand and to provide a fully customizable base that researchers can use to implement, test and compare their own approaches without having to invest considerable amounts of resources, time and knowledge.

One of the established techniques to control agents in such a simulation is through the use of Behavior Trees (BTs) [5, 11]. These are trees which are made of several decisions, which can be seen as nodes in a tree, and form a hierarchical and also a modular control structure for the agents, and which remain understandable even after the learning phase. They can also be built manually but, as with any other manual control technique, this could be very tedious if all that the agents need to do is to follow multi-unit tactics for a particular game. One way of bypassing this problem is by using a technique called Genetic Programming (GP) to build or rather to 'evolve' the structure of the BTs [15, 22]. Masek et al. [16] have shown that with sufficient learning through population level selection over many thousands of (simulated) matches, GP can be used to learn emergent combat tactics for a game called MicroRTS.

It would appear that there is at least one remaining restriction on fitness assessment in GP-based BT evolution. Current GP technology is typically able to only assess a controller (BT) at the very end of a simulation episode of play. The terminal evaluation criterion is generally either

remaining hit points or the difference in damage between the two teams. This sparse information is provided for the evolutionary process only at the termination of the episode of play, and none of the individual actions selected and executed within the episode have any direct impact on the actual fitness assigned to the overall BT. In other words, a BT with locally poor action sequences may well be assigned a moderate fitness value if its final score is still reasonably high, and a BT with relatively few suboptimal choices may be assigned a very low fitness value if it is defeated with a significantly lower score than it had at some stage during the episode of play. In summary, the current GP-based approach is only able to proceed in a slow and informationally coarse fashion. It has a coarse information base which fails to differentiate between very efficient winning strategies and those that merely succeed due to the presence of numerous suboptimal actions.

To alleviate the above issue, we develop a hybrid approach that incorporates GP-based BT evolution and uses Q-learning as the fitness-shaping mechanism within an episode. Instead of replacing the GP search, Q-learning is employed only at the terminal nodes of the BTs during each simulation episode. At the time that the BT selects an action, the value of the corresponding Q-function for that action will decide whether the action is actually executed by the agent or not. The RL-gated execution yields a per-tick quality metric of the selected actions, which is then summed up to obtain a reward for the actions selected by the evolved BT within the episode.

The Q-table is initialized at the start of each episode so knowledge of the actions in a state is not retained between episodes and cannot be used as a basis for behavioral transmission. In this sense the system can be considered a complete Baldwinian learning scenario. In this mode learning within the lifetime of an agent can affect the detail or granularity of the mapping from fitness space to behavioral space; that is, learning can increase the amount of information contained in the fitness evaluation metric, but does not alter the structure of the genotype.

The main contributions in the proposed approach are: First, the within-episode reward signal serves to transform the sparse, episodic feedback provided by the environment into a gradient of the reward function that is much denser over the space of search, thereby helping the GP to better discriminate between high and low action values for more similar BTs. Second, by punishing suboptimal action choices at the time of choice rather than retrospectively, we guide the evolutionary search toward promising action sequences that avoid unwanted behaviors such as excessive stagnation or incoherent movements. Third, our learning procedure is confined to the leaf nodes of the BT, which means that the resulting controller still benefits from the useful properties that BTs possess, specifically their hierarchical and human-readable structure that makes them practical to visually inspect and use in practice.

We investigate the behavior of our hybrid approach in controlled combat scenarios that include diverse sets of units, and the experimental results provide compelling evidence that the use of GP+RL hybrid approach yields higher peak fitness value and more clearly differentiated unit roles than the pure GP approach for BTs. Thus, even a limited form of reinforcement learning that can be considered within the episode as form of shallow shaping, seems to increase the evolution speed of GP-evolved BTs without loss of interpretability.

The remainder of this paper is organized as follows. Section 2 reviews related work on GP-based BT evolution, reinforcement learning in RTS games, and hybrid evolutionary-learning approaches. Section 3 describes the hybrid framework, including the evolutionary setup, RL module design, reward structure, and fitness evaluation procedure. Section 4 presents experimental results and analysis. Section 5 concludes the paper and outlines directions for future work.

2 Related work

Over the years, a large body of research has been carried out at the intersection of evolutionary computation, reinforcement learning and hierarchical decision-making, which is directly applicable to the field of automated tactical behavior synthesis for real-time strategy games. This section gives a brief introduction to the underlying components that form the basis of our work, namely Behavior Trees, Genetic Programming and Reinforcement Learning and finally explains in detail the motivation behind the hybrid approach we present in this paper.

Real-time strategy games (RTS) have been a prominent testbed for artificial intelligence (AI) research for many years; requiring AI agents to operate under substantial amounts of uncertainty, use a large action space and to conduct constant strategic analysis of an opponent [21, 29]. While deep reinforcement learning based systems such as AlphaStar [27] and OpenAI Five [3] have been demonstrated to have the ability to achieve a fully grandmaster level of play in popular commercial RTS games, they tend to be highly complex systems and their inner workings are not always well understood. The systems are also highly specialized, lacking the flexibility needed to be deployed in many of the scenarios where tactical decision making from an RTS would be useful. This motivated the creation of the MicroRTS environment by Ontañón et al. [20]. It is a small, fully observable environment that isolates tactical aspects from strategic part of a game and focuses solely on the tactical aspects. Over time, MicroRTS has been augmented and modified to include features such as random map scouting, different unit types and more, and is currently divided into three distinct micro-management scenarios (basic unit tracking, evolution of units and a hybrid based control flow) [10].

Behavior Trees (BTs) are currently one of the popular methods to program the decision-making of artificial agents, applied for robotics and in game AI [5, 11]. They

offer advantages such as modularity, high reusability and a clear, easily readable, hierarchical structure in contrast to, for example, finite state machines, where reactive and deliberative behaviors are not separated and analyzing, altering and drawing conclusions about decision-making of the agent is much simpler in Behavior Trees [4], an advantage that is also retained when an agent has to interact with multiple other agents [8], a phenomenon which was demonstrated with robot-based scenarios, while being largely manually constructed by the knowledge engineers based on their expertise on the subject and thus cannot scale when the complexity of the scenarios increases. In this section, we review recent research on Behavior Trees (BTs) automatic generation, as observed in the corresponding literature.

Genetic Programming (GP) – formalized by Koza [15] and later surveyed by Poli et al. [22] – provides a promising formalism for automated BT synthesis by modeling tree structures as evolvable programs. Iovino et al. [12] demonstrated that BT controllers for a robotic task can be obtained in an adaptive manner using GP in the presence of uncertainty. Styrud et al. [25] then synthesized BTs for robotic assembly tasks using GP and task planning, demonstrating the merits of structural combinations of methods for increasing the flexibility of the resulting evolved BTs in more complex, sequential decision-making scenarios.

The most relevant work to ours has been done by Masek et al. [16] where they apply GP-based BT evolution for tactical combat scenarios in MicroRTS. In their method, they evolve the structure of BTs with the help of a series of simulations. Their motivation for this method is the fact that the evolution of BTs should allow for emergent behavior of coordinated units without requiring explicit programming. The fitness of individuals is evaluated solely based on the terminal outcome signal which could be anything from the hit points difference between armies to any other desired metrics. Unfortunately, their fitness criteria are sparse and are only provided at the end of a simulation episode which makes it a very late and infrequent signal to the evolutionary search for the actions being performed in the midst of the episode. Thus, a GP-based BT evolved with optimal actions in between could potentially be provided with moderate fitness due to the fact that the overall outcome of an episode is usually adequate (even in the cases when both armies are almost eliminated, the fitness of the losing army would be smaller than that of the winner), therefore the GP cannot determine if the strategy found for the current problem is effective or just lucky. The absence of this problem has been addressed in our proposed method by introducing the within-episode feedback signal.

Reinforcement learning is an alternative approach to decision-making, where agents can learn an action policy from experience in the environment. The most basic such algorithm is known as tabular Q-learning [28]. It updates the action-values in a table based on a temporal difference between the current and next value, which is received as feedback for the current decision at each step. Deep Q-Networks (DQN) by Mnih et al. [18] showed that by applying RL to deep function approximators, agents can compete with the human level of expert performance on various control tasks. Building upon this work, other algorithms for RL in RTS have been implemented and successfully applied to full RTS games via the Gym-μRTS environment [10].

Even with their recent advances, standalone Reinforcement Learning (RL) agents for tactical game AI suffer from two problems: 1. opaque learned behaviors (policies) are difficult to understand, analyze, and reuse [11]; 2. policies learned against a particular opposing agent policy will tend to fit the particular pattern of the fixed opposing policy they were trained against and so do not generalize well to new opposing agent policies, new terrain layouts etc [24, 23]. While GP-evolved BTs offer a very different and potentially more interpretable and more flexible approach to controlling tactical game agents, it seems possible that combining an RL training stage with a BT evolved control structure may be able to blend the benefits of each approach.

The incorporation of RL signals into a BT is not a new topic in research. Dey and Child [7] proposed a framework named QL-BT (Q-learning-Based BT) based on Q-learning, which updates the action value functions for each BT node at run-time. The simulation results demonstrated that using RL to select which actions to be executed for each node in BTs can facilitate the compositional behavior model to make decisions adaptively and proved that the leaf-level learning is feasible and effective. Iovino et al. [13] applied the learned BT to collaborative multi-robot systems and demonstrated the validity of the learned BT in the multi-robot domain.

The combination of evolutionary search with within-episode learning is part of the broader class of challenges in evolution of learning, and is termed Baldwinian evolution when lifetime adaptation of an individual to its environment improves the fitness evaluation for the purpose of selection, but without the evolved parameters being heritable [2, 9]. The reason for this distinction is that in order for the individual to benefit from the evolved parameters in an evolutionary search context, they would have to be inherited, effectively replacing the learning process in the evolutionary context. Khadka and Tumer [14] have provided an example in deep RL of how evolution of population diversity and gradient based policy improvement can be beneficial, when properly decoupled. A recent survey by Sigaud [24] provides more details about hybrid evolutionary-RL approaches, and highlights that fitness-shaping (a method to define a fitness function which captures the goals of an evolutionary process in an efficient way, often by exploiting the structure of the search space) is an especially promising direction of research.

We use the theoretical framework of reward shaping that the additional rewards (or bonuses) given at every time step are designed to guide the policy search in desired ways and lead to more efficient exploration. Commonly referred to as a sparse terminal reward augmented by intermediate bonus rewards, this was shown by Ng et al. [19] that potential-

based reward shaping leads to an optimal policy in the original problem as long as a set of general conditions are satisfied. Extensions of this result have since been reported by Devlin and Kudenko [6] to the class of dynamic shaping functions where the shaping rewards may change over time. Based on these theoretical contributions, in our GP-based framework, we augment the per-tick RL rewards with an additional fitness term.

There is a clear research gap in the literature between Evolutionary Computation of BT using GPs and within-episode Reinforcement learning feedback. While several GP-based approaches for synthesis of BTs are available in the literature [16, 12], they all utilize only terminal fitness, and the lack of gradient information prevents any discrimination within the strategy at the level necessary for the problem in question. Likewise, while several approaches for integrating learning and BTs for tactical action games exist in the literature, e.g. [7], they all utilize only action or observation reward, and the resulting control decisions are only marginally improved. Finally, hybrid evolutionary-reinforcement learning approaches, e.g. [14, 24] have shown that hybrid approaches may lead to higher level of performances and more robust control policies, but these approaches have not yet been adopted for the domain of BTs evolution in tactical game domains. In our work, we fill this crucial gap by employing Q-learning within the framework of a GP-based BT evolution, in a Baldwinian approach that shapes fitness values during the search, and leads to the evolution of adaptive, interpretable and optimized BTs for a tactical action game.

3 Methodology

3.1 Overview of the hybrid framework

In this section we describe the hybrid framework employing Genetic Programming (GP) and Reinforcement Learning (RL) for evolving interpretable yet adaptable decision-making controllers for a tactical MicroRTS [20]. In this scenario, we have a decoupling between two learning components:

- the **global** evolutionary process carried out by the GP for evolving the structure of the controllers over generations (Behavior Trees, BTs);
- the **local** Q-learning [28] carried out within each gameplay episode to decide the actions of the nodes in the BT (the terminal actions) based on the instantaneous rewards given by the game at each game tick.

While the pure RL approaches lack interpretability due to the lack of transparency of the approximating functions used, and the pure GP approaches lack efficient feedback for terminal fitness functions, the hybrid approach can leverage the transparency provided by the BTs and at the same time solve the lack of feedback efficiency observed in previous works for the GP-based evolution of BTs [16].

Within each generation the evolutionary-learning loop is as follows. A population of BTs is evaluated using simulation, Q-learning updates are performed in parallel for each episode, the combined fitness is calculated from the final state and the sum of the RL-reward, and the next generation is evolved using a selection, crossover and mutation strategy. Only the BTs are evolved, and the Q-tables are reset between episodes, thus implying a Baldwinian learning model [2, 9], where the within-episode learning is for the purpose of fitness evaluation and does not bypass the evolutionary search for the optimal BTs, as the learnt information is not inherited directly.

3.2 Evolutionary framework

The evolutionary component follows the standard Genetic Programming approach [15, 22], where each individual is represented by a Behavior Tree (BT). BTs consist of two types of nodes: (i) *control nodes* (`Selector`, `Sequence`), which comply with the standard BT traversal rules [5]; (ii) *terminal nodes* (actions or conditions such as: `AttackClosestEnemy`, `MoveToClosestAllies`, `Idle`, `CheckForEnemies`) corresponding to low-level actions or conditions that are performed by each unit.

Initial population is generated by using the ramped half-and-half method [15] with full and grow initialization strategies alternated to balance the population with respect to tree depth and diversity. Parents are selected from the population by using stochastic universal sampling (SUS), which is known to produce lower variance selection pressure than the traditional fitness-proportionate roulette wheel selection method, while preserving diversity [1]. Structural variations are introduced into the individuals by using subtree crossover (SX) which swaps two randomly selected subtrees from two parent BTs, and subtree mutation (SM) which replaces a randomly selected subtree in a parent BT with a newly generated random subtree. Elitism is adopted to keep the top $\lfloor N \cdot r_e \rfloor = 10$ of best-fit individuals from the current population unchanged for the next generation, thus preventing the evolution from drifting away from the best-found solutions. Evolution always runs for the full $G = 2{,}000$ generations, as no early-termination criterion was triggered in any of the reported experiments.

3.3 Reinforcement learning module

In order to include adaptive decision making in the simulation, a tabular Q-learning process [28] is applied to each BT controller throughout each episode. Within each game tick, each unit extracts a compact state representation s_t from its local tactical state, and uses the learned action-value function to decide whether to carry out BT-suggested terminal actions.

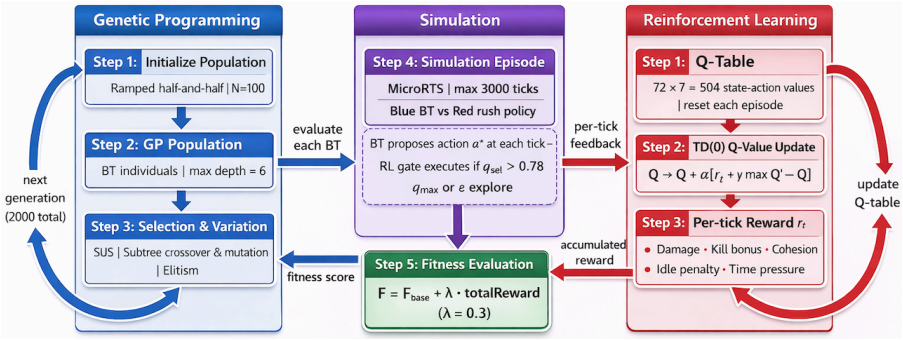

Figure 1: Overview of the hybrid GP–RL evolutionary loop. GP evolves BT structure across generations while Q-learning shapes terminal-node execution within each episode. Q-tables are reset between episodes, implementing Baldwinian adaptation.

3.3.1 State space

The state in the tabular Q-learning model for our policy is denoted by the discretized tuple:

$$s_t = \left(b_{\text{hp}},\ b_{\text{enemy}},\ b_{\text{inRange}},\ b_{\text{allyDist}},\ b_{\text{base}}\right) \tag{1}$$

The unit's health point (HP) is discretized into 3 buckets using the normalized ratio $\text{HP}/\text{HP}_{\text{max}}$: $b_{\text{hp}} \in \{0, 1, 2\}$ where the thresholds are set to 0.33 and 0.66 corresponding to low, medium and high health points respectively. The enemy-related variables are discretized into binary variables, where the value $b_{\text{enemy}} \in \{0, 1\}$ indicates whether there is a closest enemy unit within the observable field of the unit, and the value $b_{\text{inRange}} \in \{0, 1\}$ indicates whether this unit is within the attack range of the unit. The variable b_{allyDist} is discretized into 3 bins based on the Manhattan distance. The value of $b_{\text{allyDist}} = 0$ if $d \leq 2$, $b_{\text{allyDist}} = 1$ if $2 < d \leq 5$, and $b_{\text{allyDist}} = 2$ otherwise. The value of the variable b_{base} is $b_{\text{base}} \in \{0, 1\}$ to indicate the existence of a closest enemy base structure. We discretized the state space into discrete states.

$$|S| = 3 \times 2 \times 2 \times 3 \times 2 = 72 \tag{2}$$

The discrete number of states was chosen to keep the size of the Q-table manageable, while at the same time preserving the most critical information for each unit's local context.

3.3.2 Action space

The RL action space is identical to the set of BT terminal-node keys that are active during execution.

$$\mathcal{A} = \{\texttt{ACE}, \texttt{ACB}, \texttt{MTCA}, \texttt{CFE}, \texttt{CFA}, \texttt{CFB}, \texttt{Idle}\} \tag{3}$$

so that $|\mathcal{A}| = 7$. Thus, the tabular action-value function stores $|S| \times |\mathcal{A}| = 72 \times 7 = 504$ state–action values. It is important to note that the terminal nodes of the BT are of two types: *actuator terminals* (which result in the issuance of executable game commands $\{\texttt{ACE}, \texttt{ACB}, \texttt{MTCA}, \texttt{Idle}\}$) and *condition terminals* (which result in the evaluation of some predicate and returning $\{\texttt{CFE}, \texttt{CFA}, \texttt{CFB}\}$ a success or failure flag without issuing any command). See Table 1 for a detailed mapping of the action keys to the relevant BT terminal semantics.

3.3.3 RL-gated action execution

In the BT traversal phase, the terminal action a^* for the current state s_t is nominated by the RL module, which references the shared Q-table to decide whether to perform the nominated action a^* or not. *qsel* and *qmax* are respectively calculated as:

$$q_{\text{sel}} = Q(s_t,\ a^*), \qquad q_{\text{max}} = \max_{a \in \mathcal{A}} Q(s_t,\ a). \tag{4}$$

θ is the dynamic execution threshold for action a^*, which is set to:

$$\theta = 0.78 \cdot q_{\text{max}}, \tag{5}$$

qmax with a hyperparameter of 0.78 determined experimentally to obtain the optimal exploration space for BT-nominated actions while keeping enough space for the evolution search to search for other possible structures. In any case, the nominated action a^* will be performed unconditionally with exploration probability ε [26]. Otherwise the action a^* will only be performed when the value of *qsel* is:

$$q_{\text{sel}} \geq \theta. \tag{6}$$

This helps to explore the state and action space, which is very limited in the early generations when the unit should never be idle in a tactical sense at any time during the episode. If the nominated action is cancelled and the unit detects an enemy nearby, a fallback strategy will be performed in a deterministic way; in case the enemy is within attacking range, an attack command is sent to the unit, otherwise the command move toward an ally is sent. The RL-gated execution procedure is shown in Algorithm 1.

It should be noted that the deterministic fallback mechanism (lines 11–14 of Algorithm 1) is given priority over ε-greedy exploration when the gated action fails and an enemy is detected. In other words, passive actions such as `Idle` are suppressed in combat situations, regardless of the outcome of the exploration mechanism. This ensures that

Table 1: RL action keys and their corresponding BT terminal semantics.

Key	Terminal Semantics	Type
`ACE`	Attack closest enemy unit (if present)	Actuator
`ACB`	Attack closest enemy base structure (if present)	Actuator
`MTCA`	Move toward closest allied unit	Actuator
`Idle`	Remain idle unless enemy within proximity threshold	Actuator
`CFE`	Condition: at least one enemy unit exists	Condition
`CFA`	Condition: allied unit within $d \leq 2$ tiles	Condition
`CFB`	Condition: enemy base structure exists	Condition

Algorithm 1 RL-gated execution of a BT-nominated terminal action.

Input: current state s_t, BT-nominated terminal a^*, exploration rate ε
$q_{\text{sel}} \leftarrow Q(s_t,\, a^*)$
$q_{\max} \leftarrow \max_{a \in \mathcal{A}} Q(s_t,\, a)$
$\theta \leftarrow 0.78 \cdot q_{\max}$
explore $\leftarrow \big(\texttt{rand}() < \varepsilon\big)$
if *explore* **or** $q_{\text{sel}} \geq \theta$ **then**
    status $\leftarrow$ Execute(a^*)
else
    status $\leftarrow$ Failure
end if
if *status* $=$ Failure **and** enemy detected **then**
    $a_{\text{fb}} \leftarrow$ `ACE` **if** enemy in range **else** `MTCA`
    status $\leftarrow$ Execute(a_{fb})
end if
Return *status*

all units are tactically engaged at all times. In other words, the scope of exploration is restricted to non-passive actions in combat situations, which is a desirable bias.

3.4 Reward function

Q-values were updated at every game tick using the standard TD(0) update rule [28, 26]:

$$\begin{aligned} Q(s_{t-1}, a_{t-1}) \leftarrow Q(s_{t-1}, a_{t-1}) \\ +\alpha\Big[r_t + \gamma \max_{a \in \mathcal{A}} Q(s_t, a) - Q(s_{t-1}, a_{t-1})\Big], \end{aligned} \tag{7}$$

where α is the learning rate and γ is the discount factor. The per-tick reward r_t is a composite signal combining rewards for individual unit behavior and for teamwork, and was designed in accordance with potential-based reward shaping principles [19, 6] to encourage survival, damage per second to the enemy team, and to promote spatial proximity between the Heavy and Ranged units. The individual components and their respective weights can be seen in Table 2.

The local damage and self-damage terms form the core of the offense vs. defense trade-off. The team damage term then adds a cooperative aspect by giving points to individual units for the damage their teammates deal. The cohesion terms enforce that Heavy and Ranged units stay close to each other and no explicit penalty is applied to penalize the former if the latter goes too far ahead, thus preventing the common abuse of shooting ahead of a defensive shield. The idle penalty stops units from just standing around doing nothing, and the time pressure term adds a mild global pressure to keep the action moving. The per-tick rewards are then accumulated over all ticks in the episode into a single scalar `totalReward`, which is then used for the fitness calculation, as will be detailed in the next subsection.

3.5 Fitness evaluation

The fitness of the BT individuals evaluated at the end of each episode is obtained as the weighted sum of the terminal condition and the sum of the RL rewards:

$$F = F_{\text{base}} + \lambda \cdot \texttt{totalReward}, \tag{8}$$

The base fitness, F_{base}, is a measure of the terminal condition of the combat:

$$F_{\text{base}} = \Delta_{\text{HP}} + \Delta_{\text{damage}}, \tag{9}$$

where Δ_{HP} is the difference of the Blue/Red team hit points at the end of the combat and Δ_{damage} is the damage differential between the teams. The coefficient $\lambda \geq 0$, set to $\lambda = 0.3$ in our experiments and derived from a small sensitivity analysis, determines the influence of the RL reward signal; $\lambda = 0$ results in the pure GP algorithm of Masek et al. [16] that we also have as a baseline, while the effect of varying λ is discussed further in Section 4.

3.6 Simulation environment

All experiments were performed using the MicroRTS [20] simulator. MicroRTS is a lightweight, open-source real-time strategy (RTS) game simulator, providing full observability of the game state, deterministic behavior, and a large library of scripted character behaviors. MicroRTS proves to be a good testing ground for the evolutionary and hybrid approaches proposed in this paper, as they require a tightly controlled environment for fair comparisons [10]. The duration of an evaluation episode is capped at 3,000 game ticks, and episodes are always terminated and fitness computed after the earliest of these two events, regardless of the outcome of the game.

Table 2: Per-tick reward components and their contributions to r_t.

Component	Description	Weight
Local Damage	Damage inflicted on an enemy by this unit	$+1.000$
Team Damage	Damage inflicted on enemies by allied units	$+0.250$
Self Damage	Hit points lost by this unit in the current tick	-1.000
Cohesion (Spacing)	Reward when an ally is within 1–2 tiles	$+0.050$
Cohesion (Formation)	Reward when Ranged unit is positioned behind Heavy	$+0.050$
Idle Penalty	Penalty applied when unit is idle while enemies exist	-0.050
Kill Bonus	Bonus awarded when a nearby enemy unit is destroyed	$+0.500$
Time Pressure	Penalty proportional to active enemies remaining	-0.0005

The primary scenario we have for testing is a plain terrain map with no objects. For this scenario we control two different types of units on the Blue team. The first is the *Heavy* unit. A *Heavy* unit has high hit points and melee in close range. It is our front line attack unit. The second is the *Ranged*. A *Ranged* unit has low hit points and attacks at range. It will act as our support unit. For the Red team we have multiple *Light* enemy agents following a simple, deterministic rush policy where at each tick they charge toward the closest Blue team unit. The scripted enemy continuously applies pressure on the Blue team and requires our Heavy and Ranged units to undertake complex micromanagement to stay alive. We chose a plain terrain for this scenario so that we could isolate our decision-making capabilities from any path finding or terrain navigation issues that may occur in more complex scenarios, thus allowing us to more closely examine the Blue teams' tactical decisions and micromanagement required in order to protect themselves against the attacking Red team.

3.7 Parameter settings

The GP parameters were set based on the typical settings found in the literature for tree-based GP such as in [15, 22]. The population size N was set to 100 and the maximum tree depth d_{max} was set to 6, similar to that used by Masek et al. [16]. The RL parameters were set to the default values for tabular Q-learning as specified in [28, 26]. In this work, the parameters of the RL component ($\alpha = 0.1$, $\gamma = 0.9$, $\varepsilon = 0.1$) were not modified in order to investigate the effects of GP+RL integration, rather than tuning the individual hyperparameters of the RL component. The weighting factor λ was set to 0.3 after a preliminary sensitivity analysis on $\lambda \in \{0, 0.1, 0.3, 0.5\}$ that showed 0.3 to be the best trade-off between the contribution of the base combat fitness and the contribution of the RL reward. All the used parameters are summarized in Table 3.

Table 3: Parameter settings for the GP evolutionary process and RL module.

Parameter	Symbol	Value	Component
Population size	N	100	GP
Generations	G	2000	GP
Crossover probability	p_c	0.6	GP
Mutation probability	p_m	0.4	GP
Elitism ratio	r_e	0.1	GP
Maximum tree depth	d_{max}	6	GP
Learning rate	α	0.1	RL
Discount factor	γ	0.9	RL
Exploration rate	ε	0.1	RL
RL weighting factor	λ	0.3	Fitness

4 Results and discussion

4.1 Experimental setup and evaluation protocol

Experiments were conducted in the MicroRTS environment [20] on a plain, obstacle-free map to isolate tactical aspects from path planning. The Blue team consisted of a Behavior Tree-controlled Heavy unit (high HP, melee) and a Behavior Tree-controlled Ranged unit (lower HP, ranged), requiring light micro-management between them. The Red team comprises Light units that rush the nearest Blue unit each tick, applying constant pressure and forcing Blue to coordinate defense.

Each evaluation episode is limited to at most 3 000 game ticks. Parameters for this experiment are $N = 100$ BTs and $G = 2000$ generations. The two configurations to be compared under the same conditions are labeled as: 1. GP-only ($\lambda = 0$): in this case fitness is reduced to the base value F_{base} that is the terminal combat score. GP+RL hybrid ($\lambda = 0.3$): where the Baldwinian learning scheme [2, 9] is used with Q-learning as within-episode reward shaping, and without Q-values being inherited from one generation to the next. Fitness in both cases is the one defined by (8) and (9) in Section 3.

We report results for a single representative run from a set of initial training runs for each configuration, reserving multi-seed statistical replication for future work as indicated below. We note that because our focus here

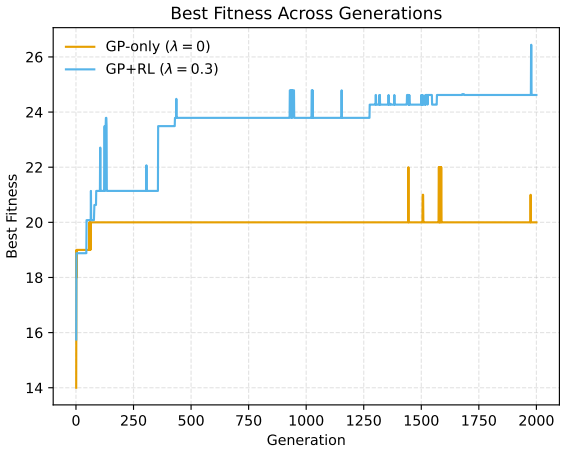

Figure 2: Best-fitness trajectory per generation for the GP-only baseline ($\lambda = 0$) and the GP+RL hybrid ($\lambda = 0.3$) over 2 000 generations. The hybrid configuration sustains improvement across a substantially longer evolutionary horizon, reaching a peak of 26.44 compared to 22.0 for GP-only, consistent with the hypothesis that within-episode reinforcement feedback densifies the fitness landscape and supports continued structural exploration.

is on understanding qualitatively the effects of including within-episode exploration through reinforcement learning as a means of shaping the evolutionary search and associated behaviors, as opposed to achieving quantitative performance guarantees on a defined metric, trajectory-level analysis of a single run per configuration is sufficient and aligns with our previous experiments [16]. These findings should thus be viewed as heuristic and aimed at motivating the direction of research explored here.

4.2 Evolutionary convergence dynamics

Figure 2 shows the best fitness of the elite BT over the 2 000 generations for both configurations. The GP-only baseline quickly found a fitness of 19.0 at generation 2 due to basic combat competencies, but was unable to show sustained directional progress in the remainder of the evolution period. The best fitness achieved by the GP-only model was 22.0 at generation 1 444, and remained almost static for the subsequent generations, where it only occasionally increased. The lack of gradient information in sparse and delayed fitness feedback, a known issue in standard GP [16, 22], means that only minor differences in within-episode tactical performance between similar BTs are available to guide the search. The fitness value of the GP-only model at generation 2,000 was 20.0. What is interesting to note is that the GP-only model never exceeded its peak fitness of 22.0 after generation 1,444, indicating that the potential of the GP-only model was exhausted with 556 generations remaining. This is a direct result of the sparse fitness gradient.

In contrast, the GP+RL hybrid exhibited a rather distinct convergence behavior. It first reached a fitness of 19.0 at generation 46, a bit later than for GP alone, due to the early-game conservatism imposed by the uninitialized Q-values, which initially contribute very little discriminative information to the fitness function and thus cause the hybrid to behave much like a noisier version of the pure GP algorithm during the early episodes. As the Q-values start to contain more knowledge about the environment and the GP is evolving better structures for the BTs (which serve as a better context for the RL gate), the per-tick reward signal contained in the fitness function became more discriminating for the structures of the BTs that choose locally optimal actions versus those that merely exploit a fluke leading to the goal. This gradual increase in the complexity of the fitness function due to the ongoing co-evolution of the GP and the RL policy allows the hybrid to remain improving for the full duration of the 2 000 generations. The hybrid eventually reached a peak fitness of 26.44 at generation 1 977 (20.2% more than the peak fitness of 22.0 — reached by GP alone). The final fitness was 24.62.

Table 4 summarizes the two slopes explicitly as fitness unit per generation for early and late phases. The early phase (generations 1–50) fitness slope for GP-only is 0.0139, and for the hybrid approach is 0.0203: therefore the hybrid model evolves faster during the first 50 generations as its slope is higher and the starting point is lower. The GP-only curve reaches very early the slope value close to zero and enters in a long near-plateau period: on the other side, the hybrid still benefits from the high performance of the RL shaping input provided through the evolutionary process, which we detail next.

4.3 Population-level diversity and stability

The mean best fitness of the GP+RL is, as expected, higher than that of the GP-only run. We have a mean best-fitness value of 23.52 for the GP+RL configuration, compared to 19.98 for the GP-only configuration. This corresponds to an increase in mean best-fitness of approximately 17.8%. Moreover, the rolling standard deviation of the best-fitness trajectory (window size 20) — a measure of local volatility within each 20-generation window — was 0.0802 for GP+RL compared to 0.0296 for GP-only. This larger local variance signaled that the hybrid continues to discover improved solutions throughout the run rather than stagnating. The validity of this claim is reinforced by the global standard deviation of the best-fit trajectory, which was calculated as 1.37 for the GP+RL algorithm in comparison with the result obtained from the GP-only algorithm, which was only 0.26. The variance was therefore 5.3× greater, showing that the hybrid algorithm was consistently finding better solutions across all 2,000 generations, as opposed to converging quickly to a near-flat trajectory. Together, both measures confirm that the RL-augmented fitness provides a landscape in which a broader variety of Behavior Tree structures receive meaningfully different fitness values, rather than being clustered at the tail of the fitness distribution with almost identical values. This is consistent

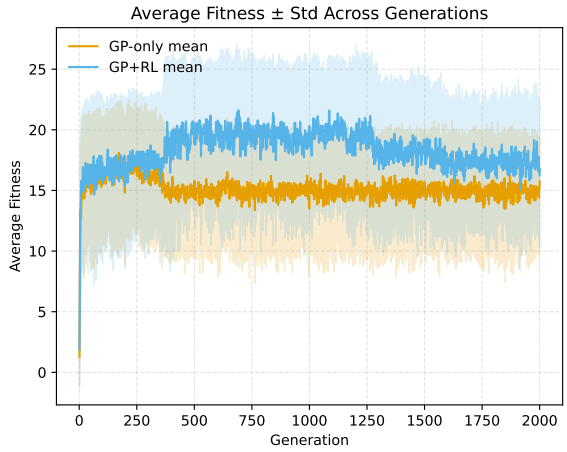

Figure 3: Mean population fitness with ± one standard deviation over 2 000 generations. The GP+RL configuration maintains a substantially higher mean population fitness (18.33 versus 15.18 for GP-only). The shaded bands represent ±1 standard deviation of the population fitness at each generation, with GP+RL showing a wider spread (mean per-generation std of 5.66 versus 4.62 for GP-only), indicating that the RL-augmented fitness landscape sustains a more diverse population throughout the evolutionary process.

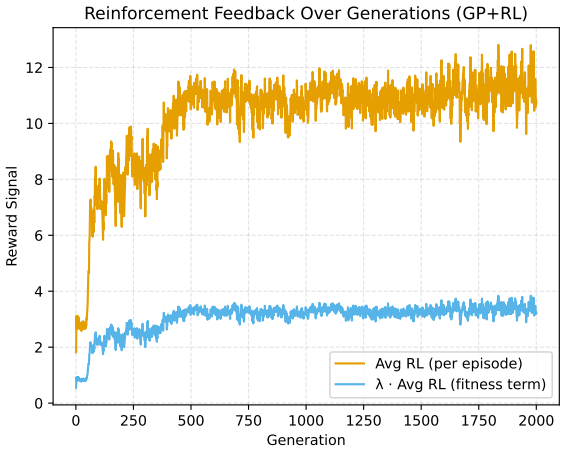

Figure 4: Evolution of the per-episode reinforcement signal in the GP+RL configuration over 2 000 generations. AvgRL increases from 1.82 (generation 1) to a global maximum value of 12.81 (generation 1 835) before settling on a final value of 10.72 at generation 2 000. The weighted fitness contribution $\lambda \cdot$ AvgRL follows the same pattern, rises to a global peak of 3.84 (generation 1 835) and settling to a final value of 3.22, confirming that the RL module provides a progressively more informative fitness-shaping signal throughout the evolutionary process.

with findings on fitness-shaping in evolutionary computation [19, 24], where a moderately high density of fitness values at intermediate points tends to enhance the evolutionary search process due to an increased exploration horizon.

4.4 Within-episode reinforcement signal quality

Figure 4 illustrates the AvgRL and the weighted fitness contribution $\lambda \cdot$ AvgRL over the 2 000 generations of the hybrid run. The AvgRL signal increased from a minimum value of 1.82 (generation 1) up to a global maximum value of 12.81 (generation 1 835) before converging to a value of 10.72 at generation 2 000. Similarly, the weighted fitness contribution $\lambda \cdot$ AvgRL increased from 0.55 (generation 1) to a global maximum value of 3.84 (generation 1 835) before converging to a value of 3.22 at generation 2 000. Therefore, it can be concluded that the Q-learning module learned increasingly better estimates of the action-values over time, due to the fact that the GP evolves structurally superior BTs and, therefore, better execution contexts for the RL gate.

These trends are strongly supported by the high correlation between AvgRL and the best-fitness trajectory ($r = 0.89$), and moderately supported by the correlation with mean population fitness ($r = 0.58$). It is likely that the reason for the higher alignment to elite fitness as opposed to average fitness is that the RL signal is used to mark and reinforce successful BTs, rather than uniformly increasing all fitnesses. This behavior is in line with the Baldwinian form of evolutionary co-adaptation [9, 14], where improved BTs lead to more conducive RL environments, which in turn produces more precise signals for evolution to act upon in the search for subsequent improvements. At the behavioral level, the increasing value of the RL signal is associated with the elimination of a number of unproductive final actions (that were not strongly penalized in the original learning session), including lingering and lack of body coordination, which are replaced by more rewarding actions such as infliction of damage and preservation of body cohesion, in accordance with the design of the reward function described in Section 3.

4.5 Effect of the reinforcement shaping weight λ

The parameter λ controls the relative importance of the within-episode RL reward in comparison to the overall fitness score. Therefore, changing the value of λ affects the relative emphasis on the terminal fight performance and the behavioral quality in the objective function for evolution. We explored two configurations, $\lambda = 0$ (GP-only) and $\lambda = 0.3$ (GP+RL hybrid). Figure 2 and Figure 3 revealed that evolution was greatly altered by setting $\lambda = 0.3$ — a relatively small adjustment to the fitness evaluation — indicating the impact of using a regularizer on the behavioral aspects of the genotypes: the evolution trajectory is greatly modified, the fitness landscape is made smoother, genotypes of diverse structures are maintained over the long-term (i.e. across the 2 000 generations), and the final fitness and mean best-fitness are strongly increased.

The rest of the parameter space would have to be swept with a different set of intermediate λ values in order to characterize the full range of transitions and to potentially shed light on the optimal point for a particular scenario. This is considered one of the major avenues for future research, in that it would allow for design of experiments that yield intelligible results on how to configure λ in a new setting without the need for experimentation. In the interim, we hope that these initial experiments suffice to make clear that the effects we are seeing are not merely trivial or artefactual in nature. Rather, we believe that they provide a robust and replicable demonstration of non-trivial impact for the predicted effects across all metrics considered.

4.6 Quantitative performance summary

Table 4 summarizes statistics of the main quantitative results from both the two experiments that ran for 2 000 generations. Note that the GP-only run achieved a maximum fitness of 22.0 at the generation 1 444 — a relatively late stage of the evolution process — and only reached a final fitness of 20.0, with a mean best-fitness of 19.98 over all the generations, along with a small standard deviation of 0.26 over the best-fitness evolution trace, indicating an almost flat trace. The GP+RL run instead achieved a maximum fitness of 26.44 at generation 1 977, a final fitness of 24.62, a mean best-fitness of 23.52 over all generations, and a standard deviation of 1.37 over the best-fitness trace. Thus the inclusion of the within-episode reinforcement learning led to an improvement in the maximum fitness of 20.2% and to an improvement in the mean best-fitness of 17.8%.

To verify the comparability of the fitness scales in the two settings, note that the population-average base combat fitness, F_{base}, at the GP+RL peak generation (1 977) is 14.63, while at the GP-only peak generation (1 444), the base combat fitness is 13.94. This observation confirms that the improvement in composite fitness has indeed resulted in an improvement in the base combat fitness and that this improvement has not been artificially induced through the inflation of the fitness scores via the RL bonus term $\lambda \cdot \texttt{totalReward}$. Notably, this confirms that the fitness gains originate from genuine combat improvements rather than mere score inflation.

4.7 Behavioral analysis of emergent tactics

While our primary focus was on the quantitative assessment of bot fitness, we noticed systematic differences in terms of bot behavior when looking at the replay videos of the simulations we ran. It appears that there are distinct patterns in the way that bots of different shapes behave, which points to some very interesting implications of the way in which individual bot learning behaviors get shaped by the reward model.

In the GP-only scenario, units generally displayed poorly coordinated and fragmented behavior (short bursts of attacking versus idling etc. and with unclear role assignments). In particular, the Heavy and Ranged units often ended up at the same positions, thus wasting the Ranged unit's shooting advantages and regularly leaving it open to the attack of the Heavy units. This occurred mainly because the rather rare fitness signal of the GP did not adequately constrain such suboptimal actions: Although the GP assesses BTs after every episode, the global fitness signal does not discriminate between individual decisions such as staying still despite being attacked, or advancing while there are still enemies within the reach of the Ranged unit, etc.

Agents using the GP+RL approach behaved in more structured and coherent ways when it comes to tactics. The Heavy unit would always advance toward the enemy to absorb incoming damage and the Ranged unit would always stay at a distance of 1-2 tiles from the enemy to constantly shoot. The structure of the behavior is enforced by the reward function (cohesion bonus and idle penalty) which motivates units to stay close to each other (formation) and heavily punishes inaction on a per-tick basis. The improvement in the agent's performance is then a matter of behavior that is more temporally coherent and more understandable in terms of strategy as opposed to simply winning more fights. This result is consistent with other work on our research line that explores the application of Reinforcement Learning to optimize the execution of BTs for a Robot Arm [12, 13]. As in those other results, the hybrid system proposed here produces as output a valid BT, maintaining full interpretability of the decisions taken by the system. This is also a desirable feature, as it avoids using the BT as just a black-box function approximator, such as what would happen if a technique like Deep Learning is applied to BTs [5, 11, 24].

4.8 Discussion

In this section, we compare our results to the state-of-the-art approaches summarized in Section 2, and analyze the reasons behind the observed improvements.

Comparison to GP-only BT evolution. For a more direct comparison, Masek et al. [16] have employed GP-based BT evolution in the MicroRTS environment using terminal fitness only. We compare our hybrid method against GP-only (reproduced here) and see that it achieves a peak fitness of 26.44 against 22.0 for the terminal-only method, thus attaining a relative improvement of 20.2%. This large difference in performance is due to the vastly different fitness landscapes that the search processes have to traverse. The terminal-only method cannot distinguish between a very efficient BT and a not so efficient one, that however wins the game by chance, due to suboptimal but lucky intermediate actions. In contrast, the within-episode RL rewards create a very dense and detailed fitness landscape. Thus, the evolutionary search is provided with a much richer gradient than the search in the terminal-only scenario, and it can take full advantage of the awarded re-

Table 4: Summary of best-fitness dynamics over 2 000 generations for both experimental configurations, computed from logged per-generation data.

Variant	Peak	Final	Mean	Std	Peak Gen.	Avg. F_{base} at peak	Slope (Gen. 1–50)
GP-only	22.00	20.00	19.98	0.26	1 444	13.94	+0.0139
GP+RL ($\lambda = 0.3$)	**26.44**	**24.62**	**23.52**	1.37	**1 977**	**14.63**	+0.0203

wards for damage and/or for keeping units together in order to traverse the search space efficiently. Idle behavior, however, is strictly penalized.

Comparison to RL-integrated BT approaches. An additional study [7] has integrated RL with BTs. However, it does not include a Baldwinian separation between withinepisode learning and across-generation evolution. Dey and Child [7] do not include any evolutionary structure search within the BT structure; therefore the number of possible BT structures is very large, but the Q-values are updated at runtime using normal RL Q-value update rules. This leads to a number of problems including the potential to by-pass the evolutionary process and lead to premature convergence as discussed in [14]. As previously mentioned, the Q-table is reset to zero at the start of each episode, therefore the within-episode learning will only serve to produce a fitness-shaping signal which will be used by the structural GP search within the evolutionary algorithm. The GP-only approach stagnated after generation 1 444; however, the sustained improvement over the 2 000 generations in the hybrid approach clearly demonstrates the Baldwinian decoupling [2, 9] between the within-episode learning and the across-generation evolution.

Comparison to hybrid evolutionary-RL approaches. The improvement of diversity and performance in continuous control domains by combining evolution with policy gradient methods has been demonstrated in work by Khadka and Tumer [14]. However, their approach targets the evolution of neural network policies, in this way lacking interpretability of the evolved controller. In our work we follow this line of research also for the evolution of BT controllers for tactical games. Here however as opposed to large-scale deep RL domains like AlphaStar [27] that achieve superior performance but are opaque, we can inspect, analyze, and even reuse the fully hierarchical and human-readable BTs evolved for tactical games by our hybrid GP+RL approach.

Novelty of the proposed approach. While several works have incorporated behavior trees (BTs) into game playing AI using Genetic Programming (GP) to evolve the structure of the trees, and within-episode reinforcement learning (RL) to learn behaviors within an episode, and even combined GP-based BT evolution with within-episode RL before, within a Baldwinian fitness inheritance framework suitable for tactical game AI, so far no work has combined GP-based BT evolution with within-episode tabular Q-learning in a Baldwinian framework for tactical game AI. Here, three main aspects of novelty are presented: (i) within-episode RL reward signals to densify the GP's sparse search space while not affecting the evolutionary search process; (ii) strict Baldwinian fitness inheritance separation by clearing the Q-tables at the start of each episode to prevent Q-values to be inherited, while allowing the GP search space to evolve in a structural way; (iii) light-weight integration, which also proves to be effective, with the search process maintaining full interpretability of the evolved BTs, and achieving both quantitative and qualitative improvements over the baseline GP search for generating functional teams of units within episodes of a tactical game, with the quantitative improvements being +20.2% peak fitness and +17.8% mean best-fitness over the baseline, and the qualitative improvements being more structured and coherent unit roles such as front-line Heavy units and support Ranged units.

Limitations and threats to validity. A main limitation of the study presented in this paper is that in each experiment a single run was performed for a given configuration. This means that it is not possible to draw statistically reliable conclusions from the results obtained. The results obtained could only be confirmed by performing multi-seed replication (i.e., ten or more independent seeds for each configuration). In addition, all experiments were performed using a single plain-terrain and against a fixed, deterministic opponent (i.e., the rush opponent). In order to draw more general conclusions, it would be necessary to perform experiments on other terrains, using different opponents, etc. Finally, the effect of the RL weighting factor λ only was studied for two values (i.e., 0 and 0.3). A more detailed study, i.e., a systematic sweep of λ over a wider range of values (e.g., 0, 0.1, 0.2, 0.3, 0.5), would be necessary in order to identify the value that leads to the best results. The main points described above are addressed as priorities for future work in Section 5.

5 Conclusion

In this work, we explored whether within-episode reinforcement learning could improve the evolution of Behavior Trees via Genetic Programming for tactical decision-making in real-time strategy games. Our motivation came

from a key limitation identified in prior work [22, 16]: standard evolutionary approaches lack within-episode feedback, making it impossible to distinguish between BTs that execute optimal sequences and those that recover from poor local choices.

To address this constraint, we proposed a hybrid approach in which tabular Q-learning [28] was used in conjunction with the BT and gated the actions toward the terminal nodes based on a learned action-value threshold and the per-tick rewards that were aggregated to form the terminal fitness score. In between the episodes, the Q-tables were reset and thus within-episode adaptations were performed in accordance with the Baldwinian learning strategy [2, 9], in which the evolved BTs are exploited to enhance the fitness evaluation, whereas the evolved action-values in the Q-tables are not encoded as genotype variables, hence not transmitted to the next generation. Overall, our hybrid approach maintains the full rigor of GP structural search, but transforms the sparse and discrete episodic fitness evaluations into a continuous dense gradient of per-tick evaluations.

Our experiments in the MicroRTS environment [20] show that the hybrid GP+RL approach consistently and significantly outperforms the base GP method across all measures. While GP alone shows slow, steady improvement over generations, the hybrid approach yields steeper fitness gains. Importantly, the RL rewards are strongly correlated with fitness, indicating that Q-learning provides meaningful guidance rather than adding noise.

At the behavioral level, the GP+RL agents displayed clear role differentiation: Heavy units charge forward to absorb damage while Ranged units maintain distance and fire continuously. In contrast, GP-only control led to more erratic behavior, with units switching unpredictably between attacking and idling. Despite these gains, the hybrid approach retains full interpretability, producing output controllers as Behavior Trees [5, 11]—a key advantage over opaque models like deep neural networks.

The results of this study are constrained by a number of limitations. First of all, our experiments give the results of a single run for each configuration, and we do not have enough statistical evidence to turn the improvements in peak fitness, mean fitness and convergence rates into firm conclusions; we recommend that multi-seed replication with at least ten independent seeds for each configuration be performed to validate the findings here reported as strong indicative evidence. Secondly, the rush opponent and plain terrain with no obstacles are deterministic and allow for the isolation of the strategic level in the simplest possible way; we will be eager to explore the behavior of the hybrid approach with more varied opponents and maps. Finally, we did not investigate the sensitivity of the results with respect to the values of evolutionary computation parameters (such as the RL weighting factor λ, the learning rate α, the discount factor γ, and the exploration rate ε) other than $\lambda = 0.3$ which was selected beforehand. The fact that within-episode learning does not accelerate the computation of the evolutionary algorithm, in spite of making each evaluation faster, is an additional observation worth mentioning: each generation is still computationally costly, because the update of the Q-table for the within-episode learning occurs at each game tick.

To address these limitations, future work will pursue multi-seed replication across at least ten independent runs per configuration to establish stronger statistical evidence. We will also conduct a thorough sensitivity analysis on λ and other RL parameters to better understand their influence on the trade-off between evolutionary and reward-based mechanisms.

Beyond validation, we plan several extensions. We will explore more complex MicroRTS scenarios, including varied terrain features and diverse opponent behaviors, to assess the generalizability of our hybrid approach. The current tabular Q-learning will be replaced with more expressive function approximation methods, such as deep Q-networks or actor-critic models [18, 17], to enhance fitness-shaping while preserving the Baldwinian separation between within-episode adaptation and across-generation evolution. We will also introduce partial observability by limiting units to local sight ranges, making the environment more realistic. Finally, we will incorporate tree-size regularization to mitigate bloat and maintain interpretability as scenario complexity increases [22].

Data Availability

The MicroRTS simulator used in this study is open-source and publicly available at `https://github.com/Farama-Foundation/MicroRTS`. The source code of the hybrid GP+RL framework, the evolved Behavior Trees, and the per-generation logs underlying Figures 2–4 and Table 4 are available from the corresponding author upon reasonable request.

References

[1] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms (ICGA)*, pages 14–21. Lawrence Erlbaum Associates, 1987.

[2] J. M. Baldwin. A new factor in evolution. *American Naturalist*, 30:441–451, 536–553, 1896.

[3] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[4] Michele Colledanchise, Diogo Almeida, and Petter Ögren. Towards blended reactive planning and acting

using behavior trees. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8839–8845. IEEE, may 2019.

[5] Michele Colledanchise and Petter Ögren. *Behavior Trees in Robotics and AI: An Introduction*. CRC Press, Boca Raton, FL, 2018.

[6] Sam Michael Devlin and Daniel Kudenko. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, volume 1, pages 433–440. IFAAMAS, 2012.

[7] Rahul Dey and Chris Child. Ql-bt: Enhancing behaviour tree design and implementation with q-learning. In *2013 IEEE Conference on Computational Intelligence in Games (CIG)*, pages 275–282, New York, NY, USA, sep 2013. IEEE.

[8] Razan Ghzouli, Thorsten Berger, Einar Broch Johnsen, Swaib Dragule, and Andrzej Wąsowski. Behavior trees in action: A study of robotics applications. In *Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering (SLE 2020)*, pages 196–209, Virtual Event, USA, nov 2020. Association for Computing Machinery (ACM).

[9] G. E. Hinton and S. J. Nowlan. How learning can guide evolution. *Complex Systems*, 1(3):495–502, 1987.

[10] S. Huang, S. Ontañon, C. Bamford, and L. Grela. Gym-μRTS: Toward affordable full game real-time strategy games research with deep reinforcement learning. In *2021 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2021.

[11] Matteo Iovino, Edvards Scukins, Jonathan Styrud, Petter Ögren, and Christian Smith. A survey of behavior trees in robotics and ai. *Robotics and Autonomous Systems*, 154:104096, aug 2022.

[12] Matteo Iovino, Jonathan Styrud, Pietro Falco, and Christian Smith. Learning behavior trees with genetic programming in unpredictable environments. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4591–4597. IEEE, jul 2021.

[13] Matteo Iovino, Jonathan Styrud, Pietro Falco, and Christian Smith. A framework for learning behavior trees in collaborative robotic applications. In *2023 IEEE 19th Conference on Automation Science and Engineering (CASE)*, pages 738–745, Kampala, Uganda, aug 2023. IEEE.

[14] Shauharda Khadka and Kagan Tumer. Evolution-guided policy gradient in reinforcement learning. *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, pages 1–12, 2018.

[15] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[16] Martin Masek, Chiou Peng Lam, Luke Kelly, and Martin Wong. Discovering optimal strategy in tactical combat scenarios through the evolution of behaviour trees. *Annals of Operations Research*, 320(2):901–936, 2023.

[17] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1928–1937, 2016.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, feb 2015.

[19] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999)*, pages 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann.

[20] S. Ontañon, N. A. Barriga, C. R. Silva, R. O. Moraes, and L. H. S. Lelis. The first MicroRTS artificial intelligence competition. *AI Magazine*, 39(1):75–78, 2018.

[21] S. Ontañon, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss. A survey of real-time strategy game AI research and competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in Games*, 5(4):293–311, 2013.

[22] R. Poli, W. B. Langdon, and N. F. McPhee. *A Field Guide to Genetic Programming*. Lulu.com, 2008.

[23] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson. The StarCraft multi-agent challenge. *Proceedings of the 18th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 2186–2188, 2019.

[24] Olivier Sigaud. Combining evolution and deep reinforcement learning for policy search: a survey. *arXiv preprint*, 2022.

[25] Jonathan Styrud, Matteo Iovino, Mikael Norrlöf, Mårten Björkman, and Christian Smith. Combining

planning and learning of behavior trees for robotic assembly. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11511–11517. IEEE, may 2022.

[26] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018.

[27] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575:350–354, 2019.

[28] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3–4):279–292, may 1992.

[29] G. N. Yannakakis and J. Togelius. *Artificial Intelligence and Games*. Springer, 2018.