ДОСЛІДЖЕННЯ МЕТОДІВ ШТУЧНОГО ІНТЕЛЕКТУ У ВІДЕОІГРАХ НА ОСНОВІ ПОВЕДІНКОВИХ ДЕРЕВ

Донець Д.С., Лєсна Н.С.

Кафедра програмної інженерії,
Харківський національний університет радіоелектроніки,
Україна

E-mail: dmytro.donets@nure.ua,
natalya.lesna@nure.ua

Abstract

The article analyzes Utility-based and Learning Behavior Trees to optimize non-player character logic, focusing on the balance between computational efficiency and adaptability. Mathematical modeling indicates that Utility systems offer superior speed for simple agents but suffer performance degradation as complexity increases, whereas Learning systems maintain stable execution times regardless of the action repertoire size. Consequently, the research establishes that Utility approaches are optimal for basic entities, while Learning architectures provide the necessary scalability for sophisticated adversaries in dynamic gaming environments.

Стрімкий розвиток індустрії комп'ютерних ігор диктує нові, підвищені вимоги до реалістичності та адаптивності неігрових персонажів (NPC). Сучасний гравець очікує від віртуального супротивника або союзника не просто набору заздалегідь прописаних скриптованих дій, а складної, непередбачуваної та контекстно-залежної поведінки. У цьому контексті класичні підходи, такі як скінченні автомати (FSM), поступово поступаються місцем більш гнучким та масштабованим архітектурам, серед яких ключову позицію займають дерева поведінки (Behavior Trees, BT). Актуальність дослідження цієї теми обумовлена нагальною необхідністю подолання проблеми комбінаторного вибуху», притаманного складним логічним системам, та пошуку оптимального балансу між продуктивністю обчислень і правдоподібністю ігрового штучного інтелекту. Дерева поведінки вже стали де-факто стандартом в індустрії, зокрема в популярних рушіях Unreal Engine та Unity, завдяки своїй модульній структурі, можливості повторного використання коду та зручності для геймдизайнерів.

У даній роботі розглядається еволюція методів побудови дерев поведінки, які класифікуються за ступенем автономності та адаптивності, починаючи від класичних статичних дерев і закінчуючи системами з елементами машинного навчання. Класичні дерева поведінки базуються на чітко визначеній ієрархії вузлів, таких як селектори, послідовності та декоратори, що дозволяє створювати детерміновану логіку, ідеальну для задач, де потрібен повний і точний контроль над діями NPC. Розвитком цієї концепції є дерева на основі корисності (Utility-based BTs) — гібридний підхід, що інтегрує концепцію Utility AI безпосередньо у структуру дерева. У таких системах замість простої бінарної логіки успіху чи невдачі вибір конкретної гілки поведінки відбувається на основі зважування пріоритетів та поточних потреб агента, що додає діям персонажа значно більше людяностіта варіативності. На вершині еволюції знаходяться дерева, що навчаються (Learning BTs), які використовують алгоритми машинного навчання, зокрема навчання з підкріпленням, для динамічної оптимізації структури дерева або параметрів окремих вузлів у реальному часі, дозволяючи агенту адаптуватися безпосередньо до стилю гри користувача. Дослідження сукупності цих методів дозволяє виявити найбільш ефективні підходи для створення імерсивного ігрового досвіду, забезпечуючи розробників потужними інструментами для побудови інтелектуальних систем нового покоління.

Порівняння Utility-based BTs (дерев поведінки на основі корисності) та Learning BTs (дерев, що навчаються) є зіставленням контрольованої раціональності та адаптивної еволюції. Головна відмінність у контексті розумності полягає в природі прийняття рішень: Utility-based BTs діють як експертна система, де розумзакладений дизайнером через формули оцінки ситуації. Це забезпечує стабільно високий рівень адекватності, оскільки NPC завжди обирає дію з найвищою корисністю (наприклад, сховатися, бо мало здоров'я). Натомість Learning BTs демонструють розумністьчерез спроможність знаходити нестандартні рішення, які дизайнер міг не передбачити. Однак, якщо Utility-based підхід гарантує логічність, то Learning BTs на етапах навчання можуть поводитися хаотично або помилково, доки не знайдуть оптимальну стратегію, що робить їхню поведінку менш передбачуваною, але потенційно більш людськоюу довгостроковій перспективі.

Щодо швидкості реакції та продуктивності, Utility-based BTs вимагають постійних обчислень. У кожному кадрі або циклі оновлення система мусить перерахувати коефіцієнти корисності для потенційних дій, щоб обрати найкращу. Чим більше варіантів дій у NPC, тим суттєвішим стає навантаження на процесор, хоча реакція на зміни ігрового світу є миттєвою. Learning BTs мають іншу специфіку: якщо модель вже навчена (offline learning), вона працює надзвичайно швидко, оскільки просто проходить по вже сформованому дереву. Однак, якщо мова йде про навчання в реальному часі (online learning), щоб адаптуватися до гравця, це створює значне обчислювальне навантаження, яке може перевищувати витрати на Utility-системи, спричиняючи затримки у прийнятті рішень на слабшому обладнанні.

Розглядаючи сценарій, де гравець діє одноманітно або не змінює тактику (наприклад, постійно атакує з однієї точки), Learning BTs мають беззаперечну перевагу. Така система з часом зрозуміє патерн гравця через механізм покарання за власні помилки та почне ефективно контрдяти (наприклад, обійде гравця з флангу). Utility-based BT у такій ситуації може стати жертвоюексплойту: якщо математична формула каже, що атака в лобмає найвищий пріоритет через близьку відстань, NPC буде раз за разом гинути однаково, доки дизайнер вручну не додасть параметри нудьгиабо штрафи за повторювані дії. Тобто Utility-система вимагає ручного налаштування для боротьби з одноманітністю, тоді як Learning-система адаптується автоматично.

У ситуації з досвідченим гравцем, який постійно змінює тактику, адаптується та діє хаотично, Utility-based BTs часто показують себе краще та стабільніше. Оскільки вони оцінюють поточний стан світу тут і зараз основі чітких критеріїв, вони миттєво реагують на нову загрозу адекватною дією, не намагаючись передбачити майбутнє, а реагуючи на факт. Learning BTs у такому динамічному хаосі можуть почати губитися або не встигати навчатися. Якщо дії гравця надто різноманітні, нейромережа чи алгоритм навчання може не знайти чіткої залежності, що призведе до неефективної або надто обережної поведінки. Для ефективної протидії такому гравцю системі навчання потрібна величезна кількість вибірок даних, яку неможливо отримати в рамках однієї короткої ігрової сесії.

T_{total} – загальний час, витрачений на прийняття рішення в одному кадрі

T_{base} – фіксований час, який виконується незалежно від прийнятого рішення.

T_{avg} – середній час потрібний на оцінку дії.

N – кількість можливих дій доступних агенту.

Тоді для оцінки загального часу справедливою буде формула:

$$T_{total} = T_{base} + k \cdot N \quad . \tag{1}$$

Розглядаються дерева середньої складності дій в умовах виконання 10, 50, 100, та 500 циклів у ігровому просторі, при статичних діях гравця, де він виконує до 3 одноманітних дій, та адаптивних діях гравця, де він виконує до 20 дій, для об'єктивного порівняння у різних ігрових ситуаціях.

Результати виконання даних симуляцій і їх загальний час представлені у табл. 1. Отримані результати розрахунків підтверджують зазначені сильні та слабкі сторони обох моделей та демонструють вигоду використання в різних випадках..

Таблиця 1. Показники часу в залежності від вибору дерева та кількості циклів

Тип дерева та кількість дій	10 циклів	50 циклів	100 Циклів	500 Циклів

Utility-based BT / 3 дії	0.53 ms	2.57 ms	5.13 ms	25.34 ms
Utility-based BT / 20 дій	3.45 ms	16.53 ms	33.19 ms	166.31 ms
Learning BT / 3 дії	1.93 ms	8.14 ms	21.93 ms	84.7 ms
Learning BT / 20 дій	2.1 ms	9.96 ms	18.93 ms	97.65 ms

Отримані дані вказують на пряму залежність швидкодії систем на основі корисності від кількості дій агента. При розширенні набору доступних операцій час обробки зростає пропорційно, що призводить до збільшення загального навантаження протягом тривалої роботи. Такий розподіл ресурсів вказує на доцільність використання цього методу для простих задач з обмеженим вибором, де відсутність складних початкових обчислень дозволяє отримати швидкий відгук. Водночас ускладнення логіки персонажа спричиняє помітне зниження продуктивності через необхідність постійного перерахунку всіх можливих варіантів.

Дерева з використанням машинного навчання демонструють сталу швидкість роботи, яка майже не змінюється при збільшенні варіативності дій. Ці системи мають фіксований поріг входу через постійні витрати на обробку даних нейромережею, що робить їх менш швидкими для простих завдань порівняно з попереднім методом. Однак ця особливість стає перевагою у сценаріях зі складними персонажами. Зі зростанням кількості циклів симуляції та розширенням тактичного арсеналу перевага переходить від простих алгоритмічних обчислень до стабільних моделей, що дозволяє підтримувати рівномірну частоту кадрів незалежно від насиченості ігрового процесу.

На підставі проведено аналізу встановлено, що доцільність використання конкретної архітектури визначається кількістю можливих дій ігрового персонажа. Системи на основі корисності виявляють високу ефективність для простих агентів завдяки швидкому відгуку, але втрачають продуктивність при ускладненні логіки через необхідність постійного перерахунку варіантів. Натомість підходи з використанням машинного навчання, маючи вищі початкові вимоги до ресурсів, зберігають сталу швидкість обробки незалежно від масштабування арсеналу дій, що робить їх оптимальним вибором для створення складних та адаптивних супротивників.

Література

1. Marcotte, Ryan, and Howard J. Hamilton. "Behavior trees for modelling artificial intelligence in games: A tutorial." *The Computer Games Journal* 6.3 (2017): C. 171-184. **DOI**: https://doi.org/10.1007/s40869-017-0040-9

2. I. Sagredo-Olivenza, P. P. Gómez-Martín, M. A. Gómez-Martín and P. A. González-Calero, "Trained Behavior Trees: Programming by Demonstration to Support AI Game Designers," in *IEEE Transactions on Games*, vol. 11, no. 1, pp. 5-14, March 2019. **DOI:** https://doi.org/10.1109/TG.2017.2771831

3. G. Robertson and I. Watson, "Building behavior trees from observations in real-time strategy games," 2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA), Madrid, Spain, 2015, pp. 1-7. DOI: https://doi.org/10.1109/INISTA.2015.7276774

4. Ji, Li Xia, and Jian Hong Ma. "Research on the Behavior of Intelligent Role in Computer Games Based on Behavior Tree." Applied Mechanics and Materials, vol. 509, Trans Tech Publications, Ltd., Feb. 2014, pp. 165–169. **DOI**: https://doi.org/10.4028/www.scientific.net/AMM.509.165

5. Tomai, E., & Flores, R. (2014). Adapting In-Game Agent Behavior by Observation of Players Using Learning Behavior Trees. Foundations of Digital Games 2014.

6. A. Ostonov and M. Moshkov, "Comparison of Complexity of Regular Versus Oblivious Decision Trees for Decision Tables With Many-Valued Decisions From Closed Classes," **DOI**: https://doi.org/10.1109/ACCESS.2025.3610956.

7 M. Nicolau, D. Perez-Liebana, M. O'Neill and A. Brabazon, "Evolutionary Behavior Tree Approaches for Navigating Platform Games," in *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 9, no. 3, pp. 227-238, Sept. 2017, **DOI:** https://doi.org/10.1109/TCIAIG.2016.2543661

8. Alberto Alvarez, Steve Dahlskog, Jose Font, and Julian Togelius. 2020. Interactive Constrained MAP-Elites: Analysis and Evaluation of the Expressiveness of the Feature Dimensions. IEEE Transactions on Games(2020). https://doi.org/10.1109/TG.2020.3046133.

9. Sekhavat, Yoones A. "Behavior trees for computer games." *International Journal on Artificial Intelligence Tools* 26.02 (2017): 1730001. **DOI**: https://doi.org/10.1142/S0218213017300010.

10. G. Florez-Puga, M. A. Gomez-Martin, P. P. Gomez-Martin, B. Diaz-Agudo and P. A. Gonzalez-Calero, "Query-Enabled Behavior Trees," in IEEE Transactions on Computational Intelligence and AI in Games, vol. 1, no. 4, pp. 298-308, Dec. 2009, **DOI**: https://doi.org/10.1109/TCIAIG.2009.2036369