УДК 004.02

О.О. МАРЧЕНКО, О.І. МАРЧЕНКО
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

СПОСІБ ДИНАМІЧНОГО РОЗПАРАЛЕЛЕННЯ ПОШУКУ В ДЕРЕВІ МЕТОДОМ МОНТЕ-КАРЛО В GRID-СИСТЕМАХ

Пошук в дереві методом Монте-Карло (MCTS) є високоефективним методом представлення інформації і пошуку рішень для задач штучного інтелекту. Особливо добре він зарекомендував себе в задачах з дуже великим степенем розгалуження пошуку, таких, як гра Го (степінь розгалуження дорівнює 250). Завдяки цьому методу відбувся прорив у створенні комп'ютерного гравця в Го, рівного по силі гри найкращим професійним гравцям.

З метою прискорення та підвищення ефективності процесу пошуку методом MCTS, його вдосконалення можливо виконувати як на алгоритмічному рівні кроків виконання MCTS, так і на рівні розпаралелення процесу пошуку, а розпаралелення, в свою чергу, як на апаратно-незалежному рівні, так і на апаратно-орієнтованому рівні. Для прискорення та покращення MCTS можуть бути використані комп'ютерні системи з різними архітектурами (багато-комп'ютерні, багато-процесорні, багато-ядерні, кластери, grid), які містять у своєму складі як CPU, так і GPU.

В даній статті виконаний аналіз відомих способів розпаралелення методу MCTS, відзначено особливості та переваги розглянутих способів. На основі цього аналізу виконано деталізацію запропонованої раніше моделі динамічного розпаралелення MCTS з використанням критеріїв типу "глибина-ширина". Ця модель узагальнює існуючі способи розпаралелення MCTS. Деталізація моделі зроблена в частині політики паралелізації MCTS. На основі деталізованої моделі запропоновано спосіб динамічного розпаралелення MCTS в grid-системах. Суть цього способу паралелізації полягає у додаванні певних оцінюючих та/або розпаралелюючих дій після кожного кроку методу MCTS на основі використання запропонованого раніше способу контролю за формою дерева. При виконанні пошуку методом MCTS запропонований спосіб паралелізації застосовується під час обробки заданої вершини дерева пошуку. Він дозволяє динамічно збільшувати/зменшувати кількість паралельних віртуальних машин (доступних обчислювальних ресурсів grid-системи) в залежності від необхідності розширювати чи поглиблювати побудову дерева пошуку, що, в свою чергу, буде призводити до прискорення процесу такого пошуку і, в результаті, до прийняття кращих рішень.

Ключові слова: задачі штучного інтелекту, дерева ігор, пошук в дереві, метод Монте-Карло, MCTS, способи розпаралелення MCTS, grid-система.

OLEKSII O. MARCHENKO, OLEKSANDR I. MARCHENKO
National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute»

DYNAMIC PARALLELIZATION TECHNIQUE FOR MONTE-CARLO TREE SEARCH IN GRID-SYSTEMS

Abstract — Monte-Carlo tree search (MCTS) method is a highly effective method for information representation and search of decisions in artificial intelligence tasks. It proves itself especially well in solving of tasks with very great branching factor such as game of Go (its branching factor equals to 250). Breakthrough in creation of a computer Go player with playing strength equivalent to strength of the best professional players happened just thankfully to this method.

With the goal to speed up and increase effectiveness of the MCTS searching process improvements of the method can be implemented both on the algorithmic level of the MCTS steps and on its parallelization level, and in its turn parallelization can be done both on the hardware-independent level and on the hardware-oriented level. Computer systems with various architectures (multi-computer, multi-processor, multi-core, clusters, grid) which include both CPUs and GPUs can be used for MCTS acceleration and improvement.

This paper deals with the analysis of known MCTS parallelization techniques and notes features and advantages of the techniques. Basing on this analysis detalization of proposed earlier a model of MCTS dynamic parallelization with usage of criteria of "depth-width" kind is done. The model generalizes existent parallelization techniques of MCTS. Detalization of the model is performed in the part of MCTS parallelization policy. On the ground of the detailed model the paper proposes a technique for MCTS dynamic parallelization in grid-systems. The core point of the parallelization technique is in certain additional estimating and parallelizing actions which are done after each step of MCTS method basing on the use of the proposed earlier tree-shape control technique. Performing search by MCTS method the proposed parallelization technique is used when a given node of a search tree is processed. The technique allows dynamical increasing/decreasing of the number of virtual machines (available computing resources of a grid-system) depending on necessity to widen or to deepen a searching tree construction, that in its turn will follow in acceleration of such searching process and in taking better decisions in result.

Keywords: artificial intelligence tasks, game trees, tree search, Monte-Carlo method, MCTS, MCTS parallelizing techniques, grid-systems.

Вступ

Прискорення обчислення задач штучного інтелекту, в яких використовуються надзвичайно великі обсяги даних і час рішення яких зараз є занадто великим, було і залишається актуальною проблемою. Апробація нових методів, способів, алгоритмів, структур даних в цій галузі виконується, як правило, на складних інтелектуальних іграх (шахи, Go), а формами подання інформації в задачах штучного інтелекту зазвичай є форми у вигляді дерев послідовних рішень.

Одним з методів штучного інтелекту, що був запропонований порівняно нещодавно і призначений для виконання швидкого пошуку правильних рішень у дереві інформації, є метод пошуку в дереві з використанням методу Монте-Карло (Monte Carlo Tree Search – MCTS) [1]. MCTS є методом пошуку оптимізованих рішень у заданій області за допомогою випадкових значень із заданого простору значень і побудови дерева пошуку за отриманими результатами. Незважаючи на відносно невеликий час існування, цей метод вже встиг суттєво вплинути на розв'язання задач штучного інтелекту.

Вдосконалення методу MCTS з метою прискорення процесу пошуку кращих рішень може бути виконано як на алгоритмічному рівні так і на рівні його розпаралелення. В той час, як ефективна паралелізація інших способів пошуку по дереву, що застосовуються в тих же задачах (наприклад, мінімаксний пошук), є проблематичною [2], то, на противагу цим способам, MCTS дозволяє зробити розпаралелення достатньо ефективно, оскільки виконує пошук по дереву напіввипадковим чином і здійснює множину моделювань подальшого розвитку подій від заданого стану, які є незалежними. На сьогодні вже були запропоновані декілька методів паралелізації методу MCTS, які дозволяють отримати непогані результати, але це, очевидно, далеко ще не всі можливості розпаралелення MCTS, особливо враховуючи велику різноманітність сучасних багатокомп'ютерних, багато-процесорних та багатоядерних обчислювальних систем.

Постановка задачі дослідження

Метою даної роботи є узагальнення процесу розпаралелення пошуку по дереву методом Монте-Карло на основі запропонованих авторами раніше моделі динамічної паралелізації [3] та способу покращення MCTS з контролем форми дерева пошуку, базуючись на критеріях типу «глибина-ширина» [4, 5], а також розробка нового способу розпаралелення на основі запропонованої моделі, який забезпечує більш ефективний розподіл наявних апаратних ресурсів для розпаралелення пошуку методом MCTS на різних етапах його роботи.

Аналіз відомих методів розпаралелення MCTS

Схема роботи методу MCTS складається з наступних чотирьох послідовних етапів [1].

Етап1: вибір.

Етап 2: розширення.

Етап 3: моделювання.

Етап 4: переобчислення.

На етапі вибору, починаючи від поточної кореневої вершини дерева пошуку і використовуючи задану політику дерева, послідовно вибираються найкращі на даний момент вершини-нащадки доки не буде досягнута довільна вершина-лист. На етапі розширення, якщо на цій вершині гра ще не закінчується, додається нова вершина, тобто відбувається спроба виконати черговий хід згідно правил гри. Цей хід із множини можливих в даному стані ходів береться згідно політики розширення, якою, як правило, є взяття випадкового ходу. На третьому етапі, починаючи від обраної нової вершини, виконується процес моделювання гри до досягнення результату (виграш/програш). Для цього можуть бути використані різні політики моделювання, але найчастіше беруть найпростішу політику, яка полягає у взятті всіх наступних ходів випадковим чином. На останньому етапі переобчислення, враховуючи отриманий результат змодельованої гри, від доданої на другому етапі вершини до кореневої вершини, у зворотному порядку виконується корекція значень атрибутів у вершинах поточної послідовності ходів згідно політики зворотного переобчислення. В якості результату однократного виконання функції пошуку методом MCTS повертається найкраща з нових вершин (з точки зору отриманого результату), отриманих під час багатократного виконання вказаних чотирьох етапів в межах заданого часу.

Як зазначалося у [6, 7], за критерієм орієнтації способу паралелізації на конкретні апаратні ресурси, чи відсутності такої орієнтації, існуючі способи паралелізації пошуку в дереві методом MCTS можуть бути поділені на апаратно-незалежні способи та апаратно-орієнтовані способи. Для розпаралелення на апаратно-незалежному рівні були запропоновані три загальних підходи: листкова паралелізація, коренева паралелізація і деревна паралелізація [1, 8, 9].

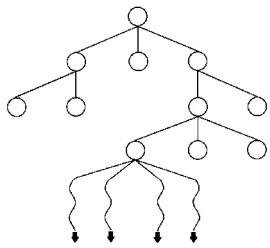

Рис.1. Листкова паралелізація

Листкова паралелізація реалізується на етапі моделювання (рис. 1) і, за визначенням даним в [8], представляє собою виконання множини одночасних моделювань від вершини-листа, яка була додана після закінчення процесу пошуку актуальної вершини, використовуючи політику дерева.

Чим більше буде виконано повторень моделювання у заданий ліміт часу, тим краща буде отримана статистика на кожній вершині-листку і, відповідно, буде обраний потенційно кращий наступний хід. Трістан Казенаве (Tristan Cazenave) та Ніколас Джоандеу (Nicolas Jouandeau) назвали таку схему паралелізацією на листках (at-the-leaves parallelization) [9], але пізніше більш широке використання отримав термін «листкова паралелізація» (leaf parallelization).

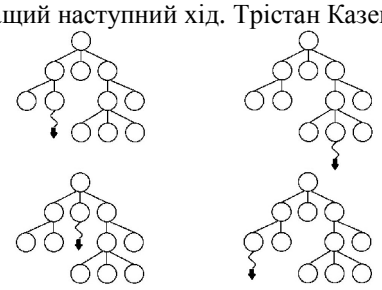

Рис.2. Коренева паралелізація

Однією з проблем листкової паралелізації є те, що різні повторення процесу моделювання від однієї і тієї ж вершини можуть зайняти різний час, і, таким чином, час виконання всіх паралельних моделювань буде дорівнювати часу найдовшого з них. Хідекі Като (Hideki Kato) і Ікуо Такеучі (Ikuo Takeuchi) [10] показали як листкова паралелізація може бути реалізована у мережі з клієнт-серверною архітектурою, в якій MCTS виконується на клієнті, а паралельні моделювання виконуються на декількох серверах.

Кореневу паралелізацію [8], яка показана на рисунку 2, іноді називають багато-деревним MCTS,

оскільки декілька дерев пошуку MCTS будуються одночасно, тобто розпаралелюються на поточному корені дерева.

Перевагою такого підходу є те, що кожен паралельний потік може бути зупинений у будь-який момент протягом заданого фіксованого часу без шкоди для коректності роботи MCTS. Треба зазначити, що найбільш популярний варіант MCTS, який має назву UCT (Upper Confidence Bounds applied to Trees), при реалізації кореневої паралелізації є алгоритмічно еквівалентним не до стандартного виду методу MCTS, яким є метод Plain UCT [8], а до його різновиду Ensemble UCT [11], який досліджували Алан Ферн (Alan Fern) та Пол Л'юїс (Paul Lewis).

Стандартним підходом для остаточного вибору наступного ходу після паралельної побудови декількох дерев пошуку від заданого кореня є знаходження суми рейтингів вершин першого рівня цих дерев і взяття вершини з найкращим сумарним рейтингом. Але Юсукі Соеїма (Yusuki Soejima), Акіхіро Кішімото (Akihiro Kishimoto) та Осаму Ватанабе (Osamu Watanabe) [12], детально проаналізувавши виконання кореневої паралелізації, показали, що проста мажоритарна схема (тобто остаточно обирається вершина, яка найчастіше була обрана в усіх паралельно побудованих деревах) дає кращі результати, ніж звичайний підхід вибору наступного ходу на основі суми рейтингів.

Трістан Казенаве (Tristan Cazenave) та Ніколас Джоандеу (Nicolas Jouandeau) описали ще два варіанти кореневої паралелізації [9], один з яких вони назвали однократною паралелізацією (single-run parallelization), а другий – багатократною паралелізацією (multiple-runs parallelization), в якій статистика ходів від кореня кожного дерева періодично надається всім паралельним потокам.

При деревній паралелізації виконується декілька одночасних ітерацій пошуку наступного ходу на одному й тому ж дереві [8]. В цьому випадку виникає класична проблема паралельного програмування, що полягає у недопущенні одночасного доступу до дерева з різних потоків. Для цього потрібно забезпечити ексклюзивний доступ до певної частини дерева MCTS кожному потоку, а також зробити так, щоб всі потоки працювали у різних областях дерева.

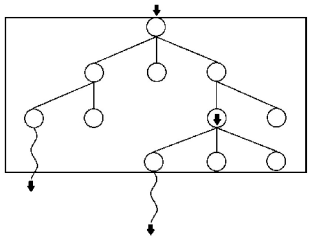

Рис.3. Деревна паралелізація з глобальним м'ютексом

В одній, запропонованій в [8], схемі деревної паралелізації використовується глобальний м'ютекс на кореневій вершині (рисунок 3). Є сенс використовувати такий підхід у випадку, якщо час виконання етапу моделювання (третього етапу MCTS) є набагато більшим, ніж час інших етапів, оскільки один потік може проходити по дереву або робити переобчислення у дереві, в той час, як інші потоки виконують моделювання.

В іншій схемі деревної паралелізації, показаній на рисунку 4, використовуються локальні м'ютекси на кожній внутрішній вершині. Ці м'ютекси блокуються і звільняються кожен раз, коли якийсь потік приходить у дану вершину.

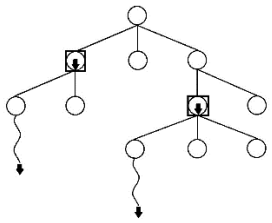

Рис.4. Деревна паралелізація з локальними м'ютексами

Одна з проблем деревної паралелізації полягає в тому, що кожен потік, скоріш за все, буде виконувати вибір наступного ходу в дереві, в основному, так само, як і інші потоки. Як один з можливих варіантів усунення такої однаковості роботи потоків, можна запропонувати призначення спеціального значення «тимчасового віртуального програшу» кожній вершині відразу після того, як вона була обрана найкращою якимсь з потоків перший раз [8]. Такий прийом примусить різні потоки вибирати різні вершини для подальшої роботи, але, в той же час, загальна схема MCTS не зміниться і, як і раніше будуть відбиратись потенційно найкращі вершини. Цей «тимчасовий віртуальний програш» повинен бути видалений безпосередньо перед кроком переобчислення для того, щоб зберегти коректність статистики.

Амін Бюркі (Amine Bourki) та інші запропонували ще один підхід, який вони назвали повільною деревною паралелізацією (slow tree parallelization) [13] і який в деяких аспектах є подібним до багатократної кореневої паралелізації [9]. При повільній деревній паралелізації зібрана у вершинах статистика періодично синхронізується між паралельно побудованими деревами, причому така синхронізація відбувається тільки на певних частинах дерева (наприклад, тільки на вершинах вище певного рівня глибини або на вершинах, які мають більшу кількість відвідувань, ніж задана). Таке рішення краще підходить для реалізації в системах з передачею повідомлень, коли взаємодія між паралельно побудованими деревами є обмеженою, наприклад, якщо паралелізація відбувається на декількох кластерах комп'ютерів. В [13] визначили, що повільна деревна паралелізація показує трохи кращі результати, ніж багатократна коренева паралелізація, незважаючи на її більш активну комунікацію між потоками. Ідея використання статистики, яка періодично синхронізується між деревами, досліджувалась також в [14].

Важливо відзначити також наступні досить очевидні особливості:

• чим більше дерев пошуку будується і оброблюється паралельно, тим краща збирається статистика і тим кращий отримується результат;

• чим більше виконується паралельних моделювань подальшої гри від обраної на етапі вибору

вершини, тим кращий отримується результат також.

Як видно з проаналізованих особливостей паралелізації MCTS, на сьогоднішній день вже було запропоновано досить багато конкретних способів паралелізації цього методу, як апаратно-незалежний, так і апаратно-орієнтованих, але в жодному з них не використовуються такі підходи як динамічний контроль за формою дерева MCTS (як воно зростає швидше, вглибину чи вширину), а також динамічний розподіл наявних апаратних ресурсів між етапами роботи MCTS, що призвело б до кращого завантаження цих ресурсів і, в результаті, до прискорення процесу пошуку і отримання кращих результатів гри.

Модель динамічної паралелізації пошуку в дереві методом Монте-Карло

Узагальнена модель динамічної паралелізації пошуку в дереві методом Монте-Карло в grid-системах вперше була запропонована авторами в [3]. З того часу вона отримала більшу деталізацію в частині опису політики паралелізації PP (див. нижче). Нагадаємо основну частину цієї моделі з метою кращого розуміння подальшого матеріалу і наведемо деталізований опис моделі в частині політики паралелізації.

Модель базується на двох підмоделях, необхідних для прийняття рішення про подальшу паралелізацію у конкретний момент часу:

STM (Search Tree Model) — модель дерева пошуку MCTS;

GAM (Grid Architecture Model) — модель архітектури grid-системи.

Модель STM дерева пошуку MCTS визначимо як

$$STM(t) = (STW(t), STD(t), NW(t), NP(t), t),$$

де STW (Sub-Tree Width) — ширина піддерева MCTS від вершини дерева, де приймається поточне рішення в момент часу t;

STD (Sub-Tree Depth) — глибина піддерева MCTS від вершини дерева, де приймається поточне рішення в момент часу t;

NW (Number of Wins) — кількість виграшів, отриманих від вершини дерева, де приймається поточне рішення в момент часу t;

NP (Number of Playouts) — кількість вже виконаних разів моделювання гри, починаючи від вершини дерева, де приймається поточне рішення в момент часу t.

Глибина STD та ширина STW піддерева MCTS обчислюються так, як описано в [4, 5].

Модель GAM архітектури grid-системи визначимо як

$$GAM(t) = (NN, Node[1...NN], t),$$

де NN (Nodes Number) — загальна кількість grid-вузлів;

Node[1...NN] — масив дескрипторів grid-вузлів, в якому кожен і-й grid-вузол визначається як

$$Node[i] = (NCC, NGC, Free(t)),$$

де NCC (Number of CPU Cores) — кількість ядер центрального процесора CPU;

NGC (Number of GPU Cores) — кількість ядер графічного процесора GPU;

Free (Flag “free/busy”) — прапорець “вільний/зайнятий” в момент часу t.

На основі цих підмоделей і загальної схеми роботи пошуку методом MCTS, модель динамічної паралелізації DPM (Dynamic Parallelization Model) пошуку в дереві методом Монте-Карло в grid-системах визначається як

$$DPM(t) = (STM(t), GAM(t), TP(t), EP(t), SP(t), BPP(t), PP(t), t),$$

де STM (Search Tree Model) — модель дерева пошуку MCTS в момент часу t;

GAM (Grid Architecture Model) — модель архітектури grid-системи в момент часу t;

TP (Tree Policy) — політика дерева (політика вибору вершини дерева для розширення) в момент часу t;

EP (Expansion Policy) — політика етапу розширення в момент часу t;

SP (Simulation Policy) — політика етапу моделювання в момент часу t;

BPP (BackPropagation Policy) — політика зворотного обчислення атрибутів дерева пошуку (фактично є певною формулою переобчислення) в момент часу t;

PP (Parallelization Policy) — політика паралелізації в момент часу t.

Політика паралелізації PP, в свою чергу, також є моделлю, яка описує види паралелізації MCTS, та їх залежність від інших параметрів описаної вище моделі.

$$ePP(t) = (RPD(t), LPD(t), TPD(t), DWCs(t), t),$$

де RPD (Root Parallelization Degree) — степінь кореневої паралелізації в момент часу t;

LPD (Leaf Parallelization Degree) — степінь листкової паралелізації в момент часу t;

TPD (Tree Parallelization Degree) — степінь деревної паралелізації в момент часу t;

DWCs (Depth-Width Criteria) — значення критеріїв DWC1 і DWC2 типу «глибина-ширина» оцінювання форми піддерева поточної вершини в момент часу t [5].

$$DWC1 = C_{DWC1} * STD / STW$$

$$DWC2 = C_{DWC2} * (P * STD – STW) / STW,$$

де C_{DWC1}, C_{DWC2}, P – коефіцієнти, які підбираються експериментальним способом або налаштовуються

методами машинного навчання.

Критерії DWCs використовуються для обчислення формул TSC1, TSC2 та TSC3 способу контролю за формою дерева MCTS, запропонованого в [5].

Параметри степені різних видів паралелізації, в свою чергу, визначаються наступним чином.

$$TPD = (STW, STD, NW, NP, NN_{free}, Nodes_{free}\ [NN], NCC_{free});$$

$$LPD = (NN_{free}, Nodes_{free}\ [NN], NCC_{free}, NGC_{free});$$

$$RPD = (STW, STD, NW, NP, NN_{free}, Nodes_{free}\ [NN], NCC_{free}),$$

де індекс «free» означає «вільні ресурси» відповідного типу, які були описані вище.

Опис способу

Накопичений досвід використання методу MCTS показав, що для одних прикладень (ігор) ефективність пошуку вище, якщо дерево зростає переважно в ширину, а для інших – в глибину [15]. Якщо ж взяти довільне нове прикладення, для якого ще не відомо, який вид дерева буде кращим, то було б доцільним виконувати динамічний контроль відношення глибина/ширина, аналізуючи зміни у ефективності пошуку і направляючи подальшу побудову дерева пошуку у потрібному напрямку.

Робота алгоритму MCTS на кроках 2 і 3 для кожної нової доданої вершини є незалежною і розпаралелюється найкраще. Тому обробка цих кроків для різних нових вершин може бути повністю виконана на окремих комп'ютерах, процесорах або процесорних ядрах. При зворотному переобчисленні атрибутів вершин на кроці 4 у випадку, якщо одночасно було змодельовано декілька шляхів гри, є можливим конфлікт при обробці однієї і тієї ж вершини, що вимагає контролю коректності спільного доступу різних паралельних процесів до спільної вершини.

Відомо [1], що збільшення кількості повторень використання вже побудованої частини дерева пошуку призводить до отримання більш глибоких гілок дерева, в той же час як збільшення кількості разів продовження пошуку від нових, ще не досліджених раніше, вершин (наступних ходів) дає побудову більш широкого дерева пошуку. Це так-звана дилема «використання-дослідження» [1]. Відповідно, більша степінь розпаралелення процесу використання призведе до більш глибокого дерева, а краще розпаралелення процесу дослідження – до більш широкого.

Але, якщо для розпаралелення процесу пошуку доступна система з фіксованою кількістю комп'ютерів/процесорів/процесорних ядер, то виконання динамічної корекції форми дерева пошуку за допомогою паралелізації наштовхується на певні проблеми. Оскільки неможливо передбачити, в який момент часу і на якій вершині потрібно буде намагатися розширювати дерево, а на якій – поглиблювати, то в заданій конкретній ситуації може не вистачити апаратних ресурсів для збільшення степені паралелізації потрібного виду (листкова, коренева, деревна) на поточному етапі MCTS. Тому, потрібно намагатись передбачати потрібну степінь паралелізації на один-декілька рівнів дерева наперед з метою більш доцільного використання апаратних ресурсів на всіх рівнях в цілому.

Спосіб паралелізації MCTS, що пропонується в даній статті, полягає у додаванні певних оцінюючих та/або розпаралелюючих дій після кожного кроку методу MCTS на основі використання запропонованого раніше способу контролю за формою дерева (TSC – Tree Shape Control) [4, 5] з метою досягнення більш високого рівня паралелізації та, як результат, покращення результативності пошуку. Перелік та спосіб виконання цих дій визначаються поточною політикою паралелізації PP.

На початку чергової ітерації MCTS перед Етапом 1, в залежності від значень поточної ширини дерева (STW) та поточної глибини дерева (STD), обчислюються критерії DWC1 та/або DWC2, а також, якщо використовується машинне навчання, виконується корекція коефіцієнтів C_{DWC1}, C_{DWC2} та P, які впливають на обчислення формули політики дерева (TP). Крім того, в залежності від наявної кількості вільних grid-вузлів (NN_{free}) та кількості ядер CPU (NCC_{free}) в них, визначається вид паралелізації верхнього рівня (деревна чи коренева) для поточної ітерації MCTS та виконується обчислення степені паралелізації у цих вершинах (TPD чи RPD), тобто кількості паралельних гілок пошуку від поточного кореня дерева MCTS.

Після вибору на Етапі 1 вершини дерева (ходу для продовження гри) згідно політики дерева (TP) та додавання нової вершини згідно політики розширення (EP) на Етапі 2, в залежності від наявної кількості вільних grid-вузлів та кількості ядер CPU (NCC_{free}), і, особливо, ядер GPU (NGC_{free}), а також від значення критеріїв DWC1 та/або DWC2 у початковій вершині даної ітерації MCTS, виконується обчислення степені листкової паралелізації (LPD) для моделювання гри від доданої вершини.

Після виконання множини паралельних ітерацій моделювання гри згідно політики моделювання (SP) на Етапі 3, виконується об'єднання результатів такого моделювання на рівні листків дерева.

Після виконання переобчислення значень атрибутів у вершинах дерева згідно політики зворотного переобчислення (BPP) на Етапі 4, виконується об'єднання результатів, отриманих під час кореневої чи деревної паралелізації різних ітерацій MCTS.

Крім того, для підвищення ефективності пошуку методом MCTS та надання апаратній платформі, на якій виконується пошук, можливості гнучкої реконфігурації, пропонується на кожному комп'ютері наявної grid-системи використовувати засоби віртуалізації.

Таким чином, запропонований спосіб дозволяє при виконанні пошуку методом MCTS, під час обробки заданої вершини дерева пошуку, динамічно збільшувати/зменшувати кількість паралельних віртуальних машин (обчислювальних ресурсів grid-системи), в залежності від необхідності розширювати чи

поглиблювати побудову дерева пошуку, що, в свою чергу, буде покращувати результативність та/або швидкість виконання пошуку.

Висновки

Пошук у дереві з використанням методу Монте-Карло (MCTS) може показувати гарні результати у багатьох областях застосування, але не у всіх.

Для одних задач MCTS виконує пошук краще, якщо форма дерева зростає переважно в ширину, а для інших – якщо в глибину.

Паралелізація пошуку в дереві з використанням методу MCTS може бути виконана згідно декількох різних підходів.

На основі аналізу методу MCTS та підходів до його паралелізації була виконана деталізація запропонованої раніше моделі динамічної паралелізації пошуку в дереві методом Монте-Карло в grid-системах.

На основі розробленої моделі паралелізації MCTS був запропонований спосіб розпаралелення процесу пошуку методом MCTS, який дозволяє динамічно контролювати критерії DWCs типу «ширина/глибина дерева» і динамічно збільшувати/зменшувати кількість паралельних потоків, отримуючи більш високу результативність і швидкість виконання пошуку.

Подальшими дослідженнями в даному напрямку можуть бути використання різних методів машинного навчання з метою динамічної корекції коефіцієнтів у критеріях DWCs, а також вплив використання цих методів на ефективність паралелізації пошуку MCTS.

Література

1. Cameron Browne. A Survey of Monte Carlo Tree Search Methods. / Cameron Browne, Edward Powley, Daniel Whitehouse, and others // IEEE Trans. on Computational Intelligence and AI in Games. – vol. 4. – no. 1. – March 2012. – P. 1-49.

2. Schaeer, J. The APHID Parallel algorithm / Schaeer, J., Brockington M. G. // Proceedings of the 8th IEEE Symposium on Parallel and Distributed Processing. – 1996. – P. 428-432.

3. Марченко О.О. Модель динамічного розпаралелення пошуку в дереві методом Монте-Карло для grid-систем. / Марченко О.О., Марченко О.І. // Системний аналіз та інформаційні технології: матеріали 19-ї Міжнародної науково-технічної конференції SAIT 2017, Київ, 22 – 25 травня 2017 р. / ННК “ІПСА” НТУУ “КПІ ім. Ігоря Сікорського”. – К.: ННК “ІПСА” НТУУ “КПІ ім. Ігоря Сікорського”. 2017., С.213-214.

4. Марченко О.О. Критерій «глибина-ширина» для контролю форми дерева пошуку при використанні методу Монте-Карло. / Марченко О.І., Марченко О.О. // Комп’ютерно-інтегровані технології: освіта, наука, виробництво. – 2016. – № 24-25. - С.42-47.

5. Oleksandr I. Marchenko. Monte-Carlo Tree Search with Tree Shape Control. / Oleksandr I. Marchenko, Oleksii O. Marchenko // 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON). Conference Proceedings. May 29 – June 2, 2017., Kyiv, Ukraine. – 2017. – P. 812-817.

6. Марченко О. І. Структура та критерії класифікації способів реалізації та покращення пошуку по дереву методом Монте-Карло / О.І. Марченко, О.О. Марченко, М.М. Орлова. // Комп’ютерно-інтегровані технології: освіта, наука, виробництво. – 2015. – № 21. – С. 51–57.

7. Марченко О.І. Класифікація способів реалізації та покращення пошуку по дереву методом Монте-Карло / О.І. Марченко, О.О. Марченко, М.М. Орлова. // Штучний інтелект. – 2016. – №2(72). – С. 59-69.

8. G. M. J.-B. Chaslot. Parallel Monte-Carlo Tree Search / G. M. J.-B. Chaslot, M. H. M. Winands, and H.J. van den Herik // Proc. Comput. And Games, LNCS 5131, Beijing, China. – 2008. P.60–71.

9. T. Cazenave. On the Parallelization of UCT / T. Cazenave and N. Jouandeau // Proc. Comput. Games Workshop, Amsterdam, Netherlands. – 2007. – P. 93–101.

10. H. Kato. Parallel Monte-Carlo Tree Search with Simulation Servers / H. Kato and I. Takeuchi // Proc. Int. Conf. Tech. Applicat. Artif. Intell., Hsinchu City, Taiwan. – 2010. – P. 491–498.

11. Fern. Ensemble Monte-Carlo Planning: An Empirical Study / A. Fern and P. Lewis // Proc. 21st Int. Conf. Automat. Plan. Sched., Freiburg, Germany. – 2011. – P. 58–65.

12. Y. Soejima. Evaluating Root Parallelization in Go / Y. Soejima, A. Kishimoto, and O. Watanabe // IEEE Trans. Comp. Intell. AI Games. – vol. 2. – no. 4. – 2010. – P. 278–287.

13. Bourki. Scalability and Parallelization of Monte-Carlo Tree Search / A. Bourki, G. M. J.-B. Chaslot, M. Coulm, V. Danjean, H. Doghmen, J.-B. Hoock, T. Herrault, A. Rimmel, F. Teytaud, O. Teytaud, P. Vayssie`re, and Z. Yu // Proc. Int. Conf. Comput. and Games, LNCS 6515, Kanazawa, Japan. – 2010. – P. 48–58.

14. S. Gelly. The Parallelization of Monte-Carlo Planning / S. Gelly, J.-B. Hoock, A. Rimmel, O. Teytaud, and Y. Kalemkarian // Proc. 5th Int. Conf. Inform. Control, Automat. and Robot., Funchal, Portugal. – 2008. – P. 244–249.

15. Hilmar Finnsson. Game-Tree Properties and MCTS Performance / Hilmar Finnsson and Yngvi Björnsson // GIGA 2011: Proceedings of the 2nd International General Game Playing Workshop. – 2011. – P. 23-30.

References

1. Cameron Browne. A Survey of Monte Carlo Tree Search Methods. / Cameron Browne, Edward Powley, Daniel Whitehouse, and others // IEEE Trans. on Computational Intelligence and AI in Games. – vol. 4. – no. 1. – March 2012. – P. 1-49.

2. Schaeer, J. The APHID Parallel algorithm / Schaeer, J., Brockington M. G. // Proceedings of the 8th IEEE Symposium on Parallel and Distributed Processing. – 1996. – P. 428-432.

3. Marchenko O.O. Model dynamichnoho rozparalelennia poshuku v derevi metodom Monte-Karlo dlia grid-system / Marchenko O.O., Marchenko O.I. // Systemnyi analiz ta informatsiini tekhnolohii: materialy 19-i Mizhnarodnoi naukovo-tekhnichnoi konferentsii SAIT 2017, Kyiv, 22 – 25 travnia 2017 r. / NNK "IPSA" NTUU "KPI im. Igoria Sikorskogo". – K.: NNK "IPSA" NTUU "KPI im. Igoria Sikorskogo". 2017., S. 213-214.

4. Marchenko O.O. Kryterii «hlybyna-shyryna» dlia kontroliu formy dereva poshuku pry vykorystanni metodu Monte-Karlo. / Marchenko O.O., Marchenko O.I. // Kompiuterno-integrovani tekhnologii: osvita, nauka, vyrobnytstvo. – 2016. – № 24-25. - S.42-47.

5. Oleksandr I. Marchenko. Monte-Carlo Tree Search with Tree Shape Control. / Oleksandr I. Marchenko, Oleksii O. Marchenko // 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON). Conference Proceedings. May 29 – June 2, 2017., Kyiv, Ukraine. – 2017. – P. 812-817.

6. Marchenko O.I. Struktura ta kryterii klasyfikatsii sposobiv realizatsii ta pokrashchennia poshuku po derevu metodom Monte-Karlo / Marchenko O.I., Marchenko O.O., Orlova M.M. // Kompiuterno-integrovani tekhnologii: osvita, nauka, vyrobnytstvo. – 2015. – № 21. – S. 51–57.

7. Marchenko O.I. Klasyfiratsiya sposobiv realizatsii ta pokrashchennia poshuku po derevu metodom Monte-Karlo / O.I. Marchenko, O.O. Marchenko, M.M. Orlova // Shtuchnyi intelekt. – 2016. – №2(72). – S. 59-69.

8. G. M. J.-B. Chaslot. Parallel Monte-Carlo Tree Search / G. M. J.-B. Chaslot, M. H. M. Winands, and H.J. van den Herik // Proc. Comput. And Games, LNCS 5131, Beijing, China. – 2008. P.60–71.

9. T. Cazenave. On the Parallelization of UCT / T. Cazenave and N. Jouandeau // Proc. Comput. Games Workshop, Amsterdam, Netherlands. – 2007. – P. 93–101.

10. H. Kato. Parallel Monte-Carlo Tree Search with Simulation Servers / H. Kato and I. Takeuchi // Proc. Int. Conf. Tech. Applicat. Artif. Intell., Hsinchu City, Taiwan. – 2010. – P. 491–498.

11. Fern. Ensemble Monte-Carlo Planning: An Empirical Study / A. Fern and P. Lewis // Proc. 21st Int. Conf. Automat. Plan. Sched., Freiburg, Germany. – 2011. – P. 58–65.

12. Y. Soejima. Evaluating Root Parallelization in Go / Y. Soejima, A. Kishimoto, and O. Watanabe // IEEE Trans. Comp. Intell. AI Games. – vol. 2. – no. 4. – 2010. – P. 278–287.

13. Bourki. Scalability and Parallelization of Monte-Carlo Tree Search / A. Bourki, G. M. J.-B. Chaslot, M. Coulm, V. Danjean, H. Doghmen, J.-B. Hoock, T. Herrault, A. Rimmel, F. Teytaud, O. Teytaud, P. Vayssie`re, and Z. Yu // Proc. Int. Conf. Comput. and Games, LNCS 6515, Kanazawa, Japan. – 2010. – P. 48–58.

14. S. Gelly. The Parallelization of Monte-Carlo Planning / S. Gelly, J.-B. Hoock, A. Rimmel, O. Teytaud, and Y. Kalemkarian // Proc. 5th Int. Conf. Inform. Control, Automat. and Robot., Funchal, Portugal. – 2008. – P. 244–249.

15. Hilmar Finnsson. Game-Tree Properties and MCTS Performance / Hilmar Finnsson and Yngvi Björnsson // GIGA 2011: Proceedings of the 2nd International General Game Playing Workshop. – 2011. – P. 23-30.

Отримана/Received : 15.9.2017 р. Надрукована/Printed :9.10.2017 р.
Стаття рецензована редакційною колегією

Рекомендовано до друку рішенням Хмельницького регіонального відділення Української технологічної академії, протокол № 3 від 13.10.2017 р.

Підп. до друку 26.10.2017 р. Ум.друк.арк. 36,51 Обл.-вид.арк. 34,74
Формат 30х42/4, папір офсетний. Друк різографією.
Наклад 100, зам. № 5713

Надруковано в типографії «ВМВ»
(Свідоцтво про видавничу діяльність ДК № 4612 від 05.09.2013)
Україна, 65069, Одеса, пр-т. Добровольського, 82а
тел. (048) 751-14-87; тел./факс 751-15-80, www.vmv.odessa.ua