АЛГОРИТМІЧНІ ТА СТРУКТУРНІ МЕХАНІКИ АЛХІМІЇ У ЧОТИРИВИМІРНИХ ІГРОВИХ ПРОСТОРАХ

Мирошниченко Р. Б.
Київський столичний університет імені Бориса Грінченка, м. Київ

ВСТУП

Розробка складних ігрових систем, що базуються на комбінуванні великої кількості елементів, є нетривіальною задачею, яка поєднує питання математичного моделювання, оптимізації обчислень та організації даних [1–2]. У межах даної роботи розглядаються ключові алгоритмічні та структурні проблеми таких систем, а також запропоновані підходи до їх розв’язання.

Актуальність

Алхімічні механіки, що передбачають створення нових сутностей шляхом змішування існуючих, формують простір станів, розмірність якого стрімко зростає зі збільшенням кількості параметрів та правил взаємодії [2–3], у наслідок чого потребує жорсткої оптимізації.

Метою дослідження є аналіз алгоритмічних і структурних проблем алхімічних систем у багатовимірних просторах та розробка підходів до їх розв’язання.

Аналіз останніх досліджень і публікацій

Проблема моделювання складних ігрових систем та механік крафтингу (зокрема алхімічних) тісно пов'язана з фундаментальними питаннями обробки великих масивів даних та обчислювальної геометрії. У роботах Кормена та Лейзерсона [1] детально розглянуто базові структури даних та алгоритми пошуку, що є основою для реалізації систем взаємодії елементів, проте вони не враховують специфіку динамічного розширення ігрового простору станів.

Важливим аспектом є проблема «прокляття розмірності» (curse of dimensionality), яку досліджували Індік та Мотвані [3]. У контексті алхімічних систем, де кожен інгредієнт може бути представлений як вектор у багатовимірному просторі ознак, пошук найближчих сусідів та визначення результату змішування стає обчислювально складним завданням. Питання ефективного хешування таких векторів та організації швидкого доступу до них у пам’яті активно обговорюються у прикладних спільнотах [8–9], де наголошується на важливості канонічного представлення даних для уникнення колізій.

Математичний апарат лінійної алгебри [4] та алгоритми зваженого середнього векторів (INFO, WMOV) [12–13] дозволяють формалізувати процес комбінування елементів не лише як дискретний набір правил, а як неперервну модель трансформації станів. Однак впровадження таких методів потребує особливої уваги до детермінованості обчислень, що

піднімає питання точності представлення чисел із плаваючою комою, ґрунтовно описані Голдбергом та стандартом IEEE 754 [6–7].

Проблеми масштабованості та збереження станів ігрових світів з великою кількістю сутностей знаходять своє відображення у дослідженнях архітектур NoSQL-сховищ, таких як LevelDB та RocksDB [10–11], які демонструють високу ефективність при роботі з ключами у вигляді впорядкованих послідовностей байтів. Крім того, Шейкер, Тогеліус та Нельсон [14] підкреслюють роль процедурної генерації контенту, яка у поєднанні з алхімічними системами дозволяє створювати унікальний ігровий досвід, але вимагає чітких механізмів обмеження простору станів для забезпечення досяжності цільових ігрових об’єктів.

Попри значну кількість робіт у суміжних галузях, питання створення цілісної архітектури для багатовимірних алхімічних систем, яка б поєднувала математичну гнучкість векторних моделей із високою продуктивністю та детермінованістю, залишається недостатньо висвітленим.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Результатом дослідження є запропонована базова модель алхімічної системи у багатовимірному просторі, що включає формалізоване представлення станів елементів, методи їх канонічного кодування та ефективного зберігання, алгоритми змішування з урахуванням пропорцій і контексту, а також механізми обмеження простору станів і забезпечення досяжності цільових елементів. Модель орієнтована на забезпечення коректності обчислень, детермінованості результатів та масштабованості при зростанні кількості елементів і правил взаємодії.

Представлення стану елемента у багатовимірному просторі

У базовій моделі кожен інгредієнт описується як сукупність параметрів, що належать до різних за природою підпросторів. Стан елемента визначається як:

$$S = (P, W, F, C),$$

де $(P \in R^4)$ – позиційний вектор, $(W \in R^4)$ – вектор ваг,

$(F \in \{0,1\}^k)$ – множина дискретних ознак $(flags)$, а (C) – контекст, що описує тимчасові властивості, отримані в процесі взаємодії [4].

Такий підхід дозволяє уникнути неконтрольованого зростання розмірності єдиного векторного простору та розділити параметри відповідно до їх природи [4–5]. Зокрема, безперервні величини обробляються окремо від дискретних, що забезпечує коректність математичних операцій.

Компактне та однозначне представлення даних

Зберігання великої кількості елементів вимагає компактного та водночас точного представлення даних. Для цього використовується канонічне кодування стану:

$$encode(S) \rightarrow K,$$

де (K) – бінарне представлення, що формується шляхом перетворення числових компонентів у їх бітове представлення відповідно до стандарту IEEE 754 з подальшим об'єднанням у єдиний буфер [6–7].

На відміну від традиційного використання хеш-функцій як ідентифікаторів, канонічне представлення виступає як повний опис стану, тоді як хеш застосовується виключно як індекс [5]:

$$h = hash(K).$$

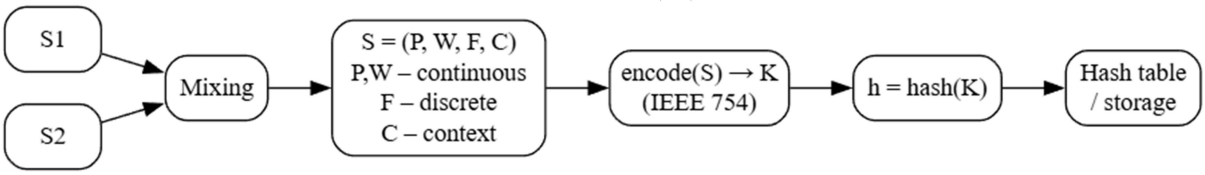

Рис.1: Компактне та однозначне представлення даних

Такий підхід [Рис.1] забезпечує, відсутність втрат точності, бієктивність, перетворення та можливість однозначного декодування [1], [8–9].

Швидкий доступ до станів без повного перебору

При роботі з великою кількістю елементів виникає задача швидкої перевірки існування вже створеного стану. Пряме порівняння всіх параметрів є обчислювально неефективним. Запропонований підхід полягає у використанні хеш-таблиць з подальшою перевіркою рівності:

1. Обчислюється хеш [1]:

$$h = hash(K),$$

2. Виконується пошук у відповідному сегменті [10–11],
3. Здійснюється точне порівняння канонічних представлень.

Такий механізм дозволяє досягти амортизованої складності доступу $(O(1))$ при збереженні коректності [1–2].

Алгоритм змішування елементів

Основна операція системи – комбінування інгредієнтів – базується на обчисленні зваженого середнього:

$$W_{new} = \frac{W_{current} * A_{current} + W_{ingredient} * A_{ingredient}}{A_{current} + A_{ingredient}},$$

$$P_{new} = \frac{P_{current} * W_{current} * A_{current} + P_{ingredient} * W_{ingredient} * A_{ingredient}}{W_{current} * A_{current} + W_{ingredient} * A_{ingredient}},$$

де (A) – кількість елементів. [12–13] (система чутлива до пропорцій)

Дискретні параметри обробляються окремо:

$$F = F_1 \cup F_2$$

Контекст оновлюється функцією переходу:

$C' = T(C_1, C_2)$,

що дозволяє враховувати порядок виконання операцій.

Процесні механіки та контекстні модифікації

Для розширення виразності системи вводиться додатковий тип об’єктів – оператори (пузирі), які змінюють контекст стану:

$$B: S \to S.$$

На відміну від інгредієнтів, такі оператори не створюють нові елементи безпосередньо, а модифікують умови подальшої обробки. Це дозволяє реалізувати такі механіки як послідовності дій, умовні рецепти та неасоціативність операцій.

Внаслідок цього система переходить від простої алгебри станів до моделі процесів [14], де результат залежить не лише від вхідних параметрів, але й від історії їх обробки.

Обмеження простору станів

Однією з ключових проблем є експоненційне зростання кількості можливих станів [2–3]. Для її вирішення застосовуються наступні методи, що дозволяють обмежити простір до скінченної, контрольованої множини:

1. Квантування – перетворення безперервних значень у дискретні, скінченні значення [6–8]:

$$x_q = round(\frac{x}{Q}),$$

де Q – крок квантування.

2. Нормалізація – приведення до канонічної форми після кожної операції,

3. Дедуплікація – уникнення створення вже існуючих станів [1–10].

Інверсна генерація елементів

Для забезпечення досяжності цільового стану використовується зворотний підхід до генерації [Рис.2].

Цільовий елемент задається як:

$P = 0.$

Подальша генерація здійснюється шляхом рекурсивного розкладу:

$$P_1 = P + d * A_1,$$
$$P_2 = P - d * A_2,$$

де (d) – довільний вектор зміщення, (A) – доля обраного відповідного інгредієнту, а (P_n) – проміжний або зберігаємий інгредієнт.

Такий підхід дозволяє побудувати дерево станів [2], у якому гарантовано існує шлях від базових елементів до цілі, а корегування вірогідностей встановлення певних елементів як проміжних та/або додавання певних умов до них дає можливість створювати нові елементні системи для кожного окремого гравця, що додає грі варіативності у подальших перепроходженнях [14].

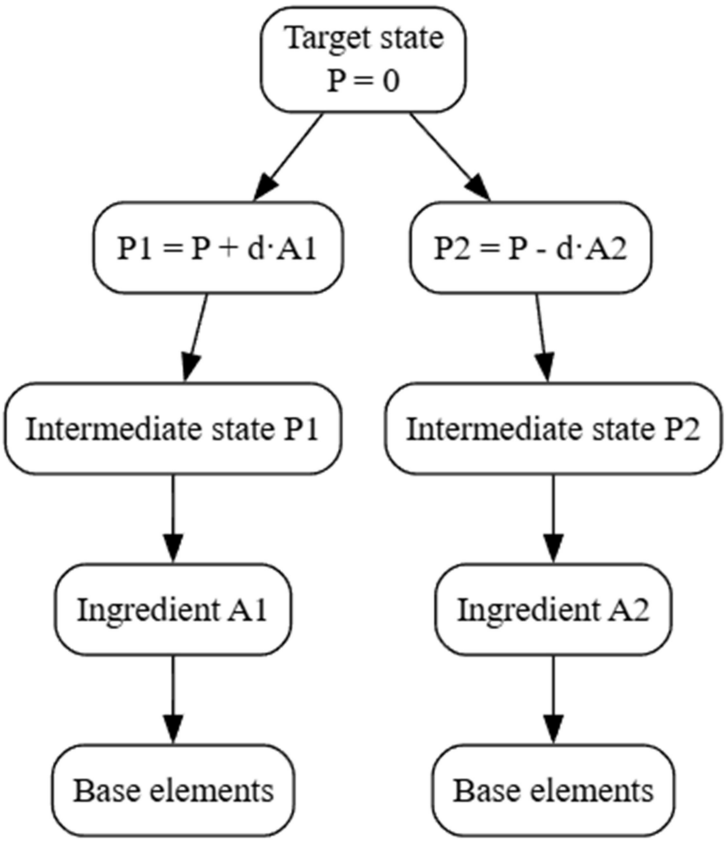

Рис. 2: Інверсна генерація станів елементів

ВИСНОВКИ

У ході проведеного дослідження було проаналізовано алгоритмічні та структурні виклики, що виникають при розробці складних алхімічних систем у багатовимірних просторах. За результатами роботи зроблено наступні висновки:

Математична формалізація

Встановлено, що представлення ігрових елементів як векторів у багатовимірному просторі ознак дозволяє гнучко описувати їхні властивості та реалізувати складні механіки змішування через апарат лінійної алгебри. Це забезпечує перехід від жорстко заданих рецептів до динамічного обчислення результатів взаємодії.

Оптимізація простору станів

Виявлено, що використання методів канонічного кодування та квантування векторів є критично важливим для стабільності системи. Це дозволяє нівелювати похибки обчислень з плаваючою комою (згідно зі стандартом IEEE 754) та забезпечити детермінованість результатів, що необхідно для мережевої синхронізації та коректного хешування даних.

Ефективність зберігання

Запропонована модель інтеграції з високоефективними сховищами (типу LevelDB/RocksDB) дозволяє системі масштабуватися до мільйонів потенційних комбінацій без втрати швидкодії. Ключовим фактором тут є перетворення багатовимірного вектора у компактний двійковий ключ.

Ігровий баланс та досяжність

Розроблені механізми обмеження простору станів дозволяють контролювати «вибухову» складність системи. Впровадження алгоритмів перевірки досяжності гарантує, що попри величезну кількість комбінацій, гравці завжди матимуть логічний шлях до створення цільових (квестових) елементів.

Практичне значення

Запропоновані підходи мінімізують ручну працю геймдизайнерів при створенні великих баз інгредієнтів і забезпечують високу продуктивність ігрового рушія при обробці складних крафтових операцій у реальному часі.

ДЖЕРЕЛА

1. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. Introduction to Algorithms. 3rd ed. Cambridge: MIT Press, 2009. 1312 p.
2. Leskovec J., Rajaraman A., Ullman J. D. Mining of Massive Datasets. 2nd ed. Cambridge: Cambridge University Press, 2014. 511 p.
3. Indyk P., Motwani R. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality // Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (STOC). 1998. P. 604–613.
4. Strang G. Linear Algebra and Its Applications. 4th ed. Boston: Cengage Learning, 2006. 576 p.
5. Knuth D. E. The Art of Computer Programming. Vol. 2: Seminumerical Algorithms. 3rd ed. Boston: Addison-Wesley, 1997. 762 p.
6. Goldberg D. What Every Computer Scientist Should Know About Floating-Point Arithmetic // ACM Computing Surveys. 1991. Vol. 23, No. 1. P. 5–48.
7. IEEE Standard for Floating-Point Arithmetic. IEEE 754-2019. New York: IEEE, 2019.
8. Stack Overflow. Good way to hash a float vector? URL:https://stackoverflow.com/questions/650175/good-way-to-hash-a-float-vector (дата звернення: 05.04.2026).
9. Stack Overflow. Hashing 2D, 3D and nD vectors. URL:https://stackoverflow.com/questions/5928725/hashing-2d-3d-and-nd-vectors (дата звернення: 05.04.2026).
10. Dean J., Ghemawat S. LevelDB Implementation. Google, 2011. URL: https://github.com/google/leveldb (дата звернення: 05.04.2026).
11. RocksDB Team. RocksDB Documentation. Facebook, 2024. URL: https://rocksdb.org (дата звернення: 05.04.2026).
12. Abdel-Basset M., Mohamed R., Mirjalili S. An efficient optimization algorithm based on weighted mean of vectors // Expert Systems with Applications. 2022. Vol. 190.
13. Zhou Y., et al. Weighted mean of vectors algorithm with neighborhood information interaction // Applied Intelligence. 2024.
14. Shaker N., Togelius J., Nelson M. Procedural Content Generation in Games. Cham: Springer, 2016. 237 p.