Алгоритмы поиска с бикритериальной оптимизацией

Проект по курсу "Эвристические методы планирования"

Угадяров Л.А.
https://github.com/ugadiarov-la-phystech-edu/hs-project

МФТИ, группа М05-006а

12 мая 2021 г.

Задача бикритериальной оптимизации

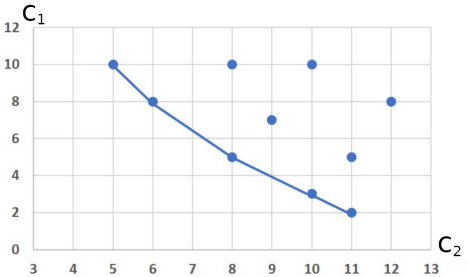
Поиск множества оптимальных решений с учётом нескольких неоднородных критериев, которые невозможно свести друг к другу.

Пусть $\mathbb{p} = (p_1, p_2)$ и $\mathbb{q} = (q_1, q_2)$ — пары вещественных чисел, тогда:

- $\mathbb{p} \prec \mathbb{q}$ ($\mathbb{p}$ доминирует $\mathbb{q}$), если $(p_1 < q_1) \wedge (p_2 \leq q_2)$ или $(p_1 = q_1) \wedge (p_2 < q_2)$
- $\mathbb{p} \leq \mathbb{q}$ ($\mathbb{p}$ слабо доминирует $\mathbb{q}$), если $(p_1 \leq q_1) \wedge (p_2 \leq q_2)$

Маршрут $\pi(s_{start}, s_{goal})$ называется парето-оптимальным $\Leftrightarrow \nexists \pi'(s_{start}, s_{goal}) : \pi' \prec \pi$.
Решение задачи бикритериальной оптимизации — множествj всех парето-оптимальных маршрутов с уникальной стоимостью.

Основная проблема адаптации A* для многокритериальных задач — низкая производительность из-за большого количества проверок доминирования.



$(10, 5) \prec (10, 8)$, $(10, 5) \not\prec (8, 6)$

Реализация алгоритмов NAMOA* и BOA*

Язык программирования: `Python 3.6`.
Используемая библиотека: `NetworkX`.
Особенности реализаций:

- Список OPEN реализован с помощью кучи. Вершина кучи — узел с лексикографически минимальным f-значением.
- Для NAMOA* реализовано ленивое удаление узлов из списка OPEN
- Если эвристическая функция неизвестна, то производится расчёт монотонной эвристической функция путём запуска алгоритма Дейкстры из целевой вершины для каждого веса - критерия
- Реализации возвращают парето-оптимальное множество весов кротчайших путей из стартовой вершины в целевую, множество родительских вершин, количество совершённых раскрытий вершин при выполнении алгоритма, время выполнения в секундах, максимальный размер списка OPEN. Также для NAMOA* возвращается максимальный размер списка OPEN без учёта вершин, помеченных как удалённые.

Проверка корректности реализаций NAMOA* и BOA*:

1. **Проверка на случайном графе из 1000 вершин**. Проверка корректности на случайном графе, который был сгенерирован с помощью библиотеки NetworkX. Параметры графа:
   - Количество вершин — 1000
   - Вероятность создания ребра между вершинами — 0.1
   - Веса рёбер – кортежи из двух элементов — случайные целые чис- ла от 1 до 1000
2. **Проверка на подграфе дорожной сети New York City (DIMACS) из 10000 вершин**.

Hernández Ulloa, C., Yeoh, W., Baier, J. A., Zhang, H., Suazo, L., & Koenig, S. (2020). A Simple and Fast Bi-Objective Search Algorithm

Сравнение производительности реализаций NAMOA* и BOA*

Производительность алгоритмов сравнивалась на 100 случайно выбранных парах вершин из графа дорожной сети New York City (DIMACS).

	Количество раскрытий	Время выполнения, с	Максимальный размер списка OPEN
Min	$4.9 * 10^1$	1.8	$2.8 * 10^1$
Mean	$4.9 * 10^5$	27.9	$2.2 * 10^4$
Max	$8.5 * 10^6$	521.7	$2.4 * 10^5$
Std	$1.3 * 10^6$	78.4	$3.6 * 10^4$

Производительность реализации BOA*

	Количество раскрытий	Время выполнения, с	Максимальный размер списка OPEN (без учёта удалённых вершин)	Максимальный размер списка OPEN
Min	$4.9 * 10^1$	1.6	$2.2 * 10^1$	$2.7 * 10^1$
Mean	$4.9 * 10^5$	189.9	$1.3 * 10^4$	$1.9 * 10^4$
Max	$8.5 * 10^6$	4236.6	$1.5 * 10^5$	$2.1 * 10^5$
Std	$1.3 * 10^6$	643.7	$2.3 * 10^4$	$3.2 * 10^4$

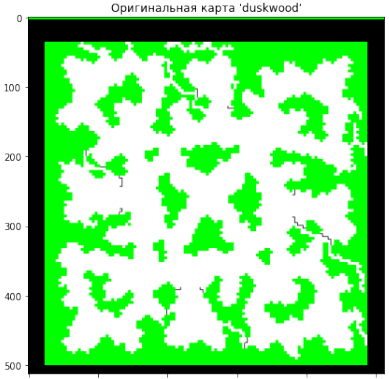
Производительность реализации NAMOA*

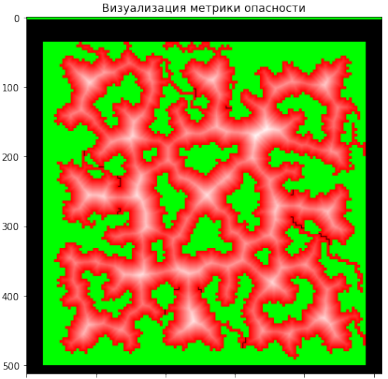
Применение BOA* для поиска множества оптимальных путей

- Граф — 8-связная карта duskwood (Moving AI Lab)
- Поиск парето-оптимального множества путей, минимизирующего длину пути и максимизирующего безопасность пути
- Безопасность проходимой клетки — расстояние до ближайшей непроходимой клетки
- Безопасность маршрута — сумма значений метрики безопосности для каждой клетки маршрута

Переход от задачи максимизации безопасности к минимизации опасности:

$$\forall v \in V \Rightarrow danger(v) = -safety(v) + \max_{v^* \in V} safety(v^*) + 1$$





Применение BOA* для поиска множества оптимальных путей

Множество парето-оптимальных маршрутов
из клетки (100, 110) в клетку (430, 400).
Полученное множество содержит 258 маршрутов.